UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros de
Telecomunicación

NONPARAMETRIC MESSAGE PASSING METHODS FOR COOPERATIVE LOCALIZATION AND TRACKING

Ph.D. Thesis
Tesis Doctoral

Vladimir Savić
Ingeniero de Electrónica

2012

DEPARTAMENTO DE SEÑALES, SISTEMAS Y RADIOCOMUNICACIONES
Escuela Técnica Superior de Ingenieros de Telecomunicación

Universidad Politécnica de Madrid

NONPARAMETRIC MESSAGE PASSING METHODS FOR COOPERATIVE LOCALIZATION AND TRACKING

TESIS DOCTORAL

Autor:

Vladimir Savić

Ingeniero de Electrónica

Director:

Santiago Zazo Bello

Profesor titular del Dpto. de Señales, Sistemas y Radiocomunicaciones
Universidad Politécnica de Madrid

2012

TESIS DOCTORAL

NONPARAMETRIC MESSAGE PASSING METHODS FOR COOPERATIVE LOCALIZATION AND TRACKING

AUTOR: Vladimir Savić
DIRECTOR: Santiago Zazo Bello

Tribunal nombrado por el Mgfco. y Excmo. Sr. Rector de la Universidad Politécnica de Madrid, el día ___ de _____ de 2012.

PRESIDENTE: Narciso García Santos

SECRETARIO: Jesús Grajal de la Fuente

VOCAL: Antonio Artés Rodríguez

VOCAL: Joaquín Míguez Arenas

VOCAL: Lennart Svensson

SUPLENTE: Tomas McKelvey

SUPLENTE: Jesús Pérez Arriaga

Realizado el acto de defensa y lectura de Tesis el día ___ de _____ de 2012.

En la E.T.S. de Ingenieros de Telecomunicación.

Calificación:

EL PRESIDENTE LOS VOCALES

EL SECRETARIO

To my family

Acknowledgements

This Ph.D thesis would not have been possible without the support of many people. First of all, I would like to thank my supervisor Santiago Zazo Bello for his help and many useful advices. I also appreciate that he gave me a chance to participate in many international conferences, and to perform a part of my research at another universities.

Furthermore, I would like to thank Petar M. Djurić (Stony Brook University) for supervising me during my visit, and for providing me the access to the laboratory. I also thank Akshay Athalye (Stony Brook University), and Miodrag Bolić (University of Ottawa) for useful advices, and their support for performing the experiments.

Many thanks to Henk Wymeersch (Chalmers University) and Federico Penna (Politecnico di Torino) for their collaboration, which resulted in the novel algorithm useful for many applications, including the topic of this thesis. Moreover, I thank Henk Wymeersch for supervising me during my visit at Chalmers University. I also thank Lennart Svensson (Chalmers University) for the useful proposals on how to reduce the complexity of the proposed algorithms.

Many thanks to my colleagues at "Grupo de Aplicaciones de Procesado de Señal" (GAPS), especially Adrián Población, Benjamin Béjar, Igor Arambasić, Ivana Raos, Mariano García, Nelson Dopico, Pavle Belanović, Sergio Valcarcel. Moreover, I thank to colleague from "Grupo de Microondas y Radar" (GMR), Ángel García Fernández. I especially appreciate their advices that improved my research results, and their help to solve some administrative problems.

I would like to thank Bernard Henri Fleury (Aalborg University), Jesús Grajal (Universidad Politecnica de Madrid) and Ronald Raulefs (German Aerospace Center - DLR) for taking their time to review my thesis.

Finally, I also thank my family and all friends. Special thanks to my parents, Jovan and Olga, for their enormous support during my stay in Spain.

Abstract

The objective of this thesis is the development of cooperative localization and tracking algorithms using nonparametric message passing techniques. In contrast to the most well-known techniques, the goal is to estimate the posterior probability density function (PDF) of the position of each sensor. This problem can be solved using Bayesian approach, but it is intractable in general case. Nevertheless, the particle-based approximation (via nonparametric representation), and an appropriate factorization of the joint PDFs (using message passing methods), make Bayesian approach acceptable for inference in sensor networks. The well-known method for this problem, nonparametric belief propagation (NBP), can lead to inaccurate beliefs and possible non-convergence in loopy networks. Therefore, we propose four novel algorithms which alleviate these problems: nonparametric generalized belief propagation (NGBP) based on junction tree (NGBP-JT), NGBP based on pseudo-junction tree (NGBP-PJT), NBP based on spanning trees (NBP-ST), and uniformly-reweighted NBP (URW-NBP). We also extend NBP for cooperative localization in mobile networks. In contrast to the previous methods, we use an optional smoothing, provide a novel communication protocol, and increase the efficiency of the sampling techniques. Moreover, we propose novel algorithms for distributed tracking, in which the goal is to track the passive object which cannot locate itself. In particular, we develop distributed particle filtering (DPF) based on three asynchronous belief consensus (BC) algorithms: standard belief consensus (SBC), broadcast gossip (BG), and belief propagation (BP). Finally, the last part of this thesis includes the experimental analysis of some of the proposed algorithms, in which we found that the results based on real measurements are very similar with the results based on theoretical models.

Resumen

El objetivo de esta tesis es el desarrollo de los algoritmos para posicionamiento y seguimiento cooperativo mediante técnicas no paramétricas de paso de mensajes. En contraste con la mayoría de técnicas bien conocidas, el objetivo es estimar la función densidad de probabilidad posterior de la posición de cada sensor. Este problema se puede resolver mediante técnica bayesiana, pero es insoluble en el caso general. Sin embargo, se puede resolver utilizando la aproximación basada en partículas (a través de la representación no paramétrica), y una factorización apropiada de las funciones densidad de probabilidad conjunta (utilizando métodos de paso de mensajes). Método bien conocido para este problema, *nonparametric belief propagation* (NBP), puede causar certezas inexactas y posible falta de convergencia en las redes con bucles. Por lo tanto, proponemos cuatro nuevos algoritmos que pueden resolver estos problemas: *nonparametric generalized belief propagation* (NGBP) basado en *junction tree* (NGBP-JT), NGBP basado en *pseudo-junction tree* (NGBP-PJT), NBP basado en *spanning trees* (NBP-ST), y *uniformly-reweighted* NBP (URW-NBP). También proponemos extension de NBP para posicionamiento cooperativo en redes móviles. En contraste con los métodos anteriores, enviamos un mensaje opcional desde el futuro al presente, proponemos el nuevo protocolo para comunicación, y aumentamos eficaz de técnicas de muestreo. Además, proponemos nuevos algoritmos para el seguimiento distributivo, en el que el objetivo es realizar el seguimiento del objeto pasivo que no puede ubicarse. En particular, desarrollomos filtros de particulas distributivas (DPF) basados en tres *belief consensus* (BC) algoritmos: *standard belief consensus* (SBC), *broadcast gossip* (BG), y *belief propagation* (BP). Finalmente, la última parte de esta tesis incluye el análisis experimental de algunos de los algoritmos propuestos, en el que se encontró que los resultados basados en medidas reales son muy similares a los resultados basados en modelos teóricos.

Contents

List of Figures

List of Acronyms

AFL	Anchor-free localization
ANBP	Auxiliary nonparametric belief propagation
AOA	Angle of arrival
APF	Auxiliary particle filtering
BC	Belief consensus
BFS	Breadth first search
BG	Broadcast gossip
BP	Belief propagation
CAB	Concentric anchor beacon
CDF	Cumulative distribution function
CPF	Centralized particle filtering
DOM	Direction of movement
DPF	Distributed particle filtering
DV	Distance-vector
GBP	Generalized belief propagation
GPS	Global positioning system
JT	Junction tree
KDE	Kernel density estimate
KF	Kalman filter
KLD	Kullback-Leibler (KL) divergence

LOS	Line-of-sight
MAP	Maximum a posteriori
MC	Max-consensus
MDS	Multi-dimensional scaling
MIS	Mixture importance sampling
ML	Maximum likelihood
MMSE	Minimum mean square estimate
NBBP	Nonparametric boxed belief propagation
NBP	Nonparametric belief propagation
NCPF	Non-centralized particle filtering
NGBP	Nonparametric generalized belief propagation
NLOS	Non-line-of-sight
PDF	Probability density function
PF	Particle filtering
PJT	Pseudo-junction tree
PMC	Population Monte Carlo
RFID	Radio-frequency identification
RIP	Running intersection property
RMSE	Root-mean-square (RMS) error
RP	Reference particles
RSS	Received signal strength
SBC	Standard belief consensus
SIR	Sample-importance-resampling
ST	Spanning tree
TDOA	Time difference of arrival
TG	Thin graph

TOA	Time of arrival
TRW	Tree-reweighted
UHF	Ultra-high frequency
URW	Uniformly-reweighted
UWB	Ultra-wide band
WC	Weighted centroid
WLAN	Wireless local area networks
WSN	Wireless sensor network

Chapter 1

Introduction

1.1 Thesis objectives

In this thesis, we develop novel cooperative localization and tracking algorithms, for static and mobile networks, using nonparametric message passing techniques. In contrast to the most well-known techniques, the goal is to estimate posterior probability density function (PDF) of the position of each sensor. This problem can be solved using Bayesian approach, but it is intractable in general case. Nevertheless, the particle-based approximation (via nonparametric representation), and an appropriate factorization of the joint PDFs (using message passing methods), make Bayesian approach acceptable for inference in sensor networks. Extensions of well-known method for this problem, nonparametric belief propagation (NBP), are the main topic of this thesis. There are four main objectives:

- Development of novel message passing methods for cooperative localization in loopy networks. The goal is to improve performance of standard NBP, which can lead to inaccurate beliefs and possible non-convergence in loopy networks.
- Development of novel NBP-based algorithms for cooperative localization in mobile networks. The goal is to use smoothing nearly in real time, decrease the communication cost, and increase the efficiency of sampling techniques.
- Development of novel belief consensus methods for distributed tracking of the passive object. The goal is to use fastest consensus method, and to allow the use of all parametric and nonparametric likelihood functions.
- Experimental analysis in indoor environment of some of the proposed algorithms for static and mobile networks. To that end, we use IRIS wireless motes, and semi-passive Radio-Frequency IDentification (RFID) system.

1.2 Outline of the thesis

The rest of this thesis is organized as follows:

- Chapter 2 reviews cooperative (multi-hop) localization techniques, in which small number of sensors, called anchor nodes, obtain their coordinates via Global positioning system (GPS) or by installing them at points with known coordinates, and the rest, unknown nodes, must determine their own coordinates using the anchor's positions and measured inter-sensor distances. Since the sensors are usually energy-conserving devices, i.e., without energy necessary for long-range communication, they have available only the noisy measurements of the distance to several neighboring nodes. In particular, we describe standard measurement techniques, deterministic localization methods (distance-based and connectivity-based), localization using angle of arrival (AOA), and general framework for probabilistic localization.

- Chapter 3 addresses static positioning using improved message passing methods. We first describe and analyse standard techniques, BP and NBP. Then, we propose nonparametric boxed belief propagation (NBBP), in which we added the bounded boxes to constraint the area from which the particles are drawn. Since all of these methods (BP, NBP, NBBP) can lead to inaccurate beliefs and possible non-convergence in loopy networks, we propose four improved message-passing methods: nonparametric generalized belief propagation (NGBP) based on junction tree (NGBP-JT), NGBP based on pseudo-junction tree (NGBP-PJT), NBP based on spanning trees (NBP-ST), and uniformly-reweighted NBP (URW-NBP).

- Chapter 4 addresses two important problems: cooperative localization in mobile networks, and distributed tracking of the passive object. For the first problem, we extend NBP described in Chapter 3. In contrast to previous methods, we send optional message from the future to present using only 1-leg smoothing, and solve two important problems of the standard NBP method: decrease the communication cost, and increase the efficiency of the sampling techniques. For the second problem, distributed tracking, the goal is to track the passive object which cannot locate itself (in contrast to cooperative localization in mobile networks). Since the current state-of-the-art methods do not use fastest consensus algorithms, and also most of them cannot handle all parametric and nonparametric likelihood functions, we propose novel general framework for distribute target tracking. In particular, we propose distributed particle filtering (DPF) based on three asynchronous belief consensus (BC) al-

gorithms: standard belief consensus (SBC), broadcast gossip (BG), and belief propagation (BP).

- Chapter 5 includes the experimental analysis of some of the proposed algorithms in previous chapters. We analyse cooperative localization based on NBP, and NBP-ST, described in Chapter 3, and distributed tracking using DPF based on BC algorithms, described in Chapter 4. Experimental analysis of NBP and NBP-ST cooperative localization methods is performed using received signal strength (RSS) data obtained in indoor environment. For these experiments, Crossbow's IRIS wireless motes has been used, which is fully compatible with ZigBee/IEEE802.15.4 standard. Distributed tracking has been analysed using semi-passive (sensatag-based) RFID system.
- Chapter 6 includes the conclusions and suggestions for the future work.

1.3 List of publications

The main results of this thesis have been published in following international journals and conferences:

1. V. Savic and S. Zazo, "Reducing communication overhead for cooperative localization using nonparametric belief propagation," in *IEEE Wireless Communications Letters*, 2012.
2. H. Wymeersch, F. Penna, and V. Savic, "Uniformly reweighted belief propagation for estimation and detection in wireless networks," in *IEEE Trans. on Wireless Communications*, 2012.
3. V. Savic and S. Zazo, "Belief propagation techniques for cooperative localization in wireless sensor networks," in *Position Location - Theory, Practice and Advances: A Handbook for Engineers and Academics*, Wiley, 2011.
4. V. Savic, A. Athalye, M. Bolic and P. M. Djuric, "Particle filtering for indoor RFID tag tracking," in *IEEE Proc. of Statistical Signal Processing (SSP)*, Nice, France, June 2011.
5. H. Wymeersch, F. Penna, and V. Savic, "Uniformly reweighted belief propagation: A factor graph approach," in *IEEE Proc. of Intl. Symposium on Information Theory (ISIT)*, St. Petersburg, Russia, July 2011.
6. F. Penna, H. Wymeersch and V. Savic, "Uniformly reweighted belief propagation for distributed Bayesian hypothesis testing," in *IEEE Proc. of Statistical Signal Processing (SSP)*, Nice, France, June 2011.

7. V. Savic , H. Wymeersch , F. Penna and S. Zazo, "Optimized edge appearance probability for cooperative localization based on tree-reweighted nonparametric belief propagation," in *Proc. of IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 3028-3031, Prague, Czech Rep., May 2011.

8. A. Athalye, V. Savic, M. Bolic and P. M. Djuric, "A radio frequency identification system for accurate indoor localization," in *Proc. of IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1777-1780, Prague, Czech Rep., May 2011.

9. V. Savic, A. Poblacion, S. Zazo, and M. Garcia, "Indoor positioning using nonparametric belief propagation based on spanning trees," *EURASIP Journal on Wireless Communications and Networking*, 2010.

10. V. Savic and S. Zazo, "Nonparametric belief propagation based on spanning trees for cooperative localization in wireless sensor networks," in *IEEE Proc. of VTC-Fall*, Ottawa, Canada, Sept. 2010.

11. V. Savic and S. Zazo, "Pseudo-junction tree method for cooperative localization in wireless sensor networks," in *IEEE Proc. of Information Fusion*, Edinburgh, UK, July 2010.

12. V. Savic, A. Poblacion, S. Zazo, and M. Garcia, "An Experimental study of RSS-based indoor localization using nonparametric belief propagation based on spanning trees," in *IEEE Proc. of the Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pp. 238-243, Venice, Italy, July 2010.

13. V. Savic and S. Zazo, "Sensor localization using nonparametric generalized belief propagation in network with loops," in *IEEE Proc. of Information Fusion*, pp. 1966-1973, Seattle, USA, July 2009.

14. V. Savic and S. Zazo, "Sensor localization using generalized belief propagation in network with loops," in *Proc. of the 17th European Signal Processing Conference - EUSIPCO*, pp. 75-79, Glasgow, UK, August 2009.

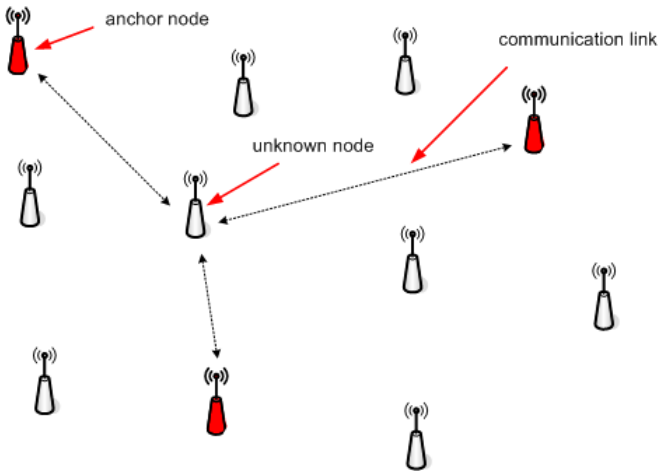
15. V. Savic and S. Zazo, "Nonparametric boxed belief propagation for localization in wireless sensor networks," in *IEEE Proc. of the Third International Conference on Sensor Technologies and Applications (SENSORCOMM)*, pp. 520-525, Athens, Greece, June 2009.

Chapter 2

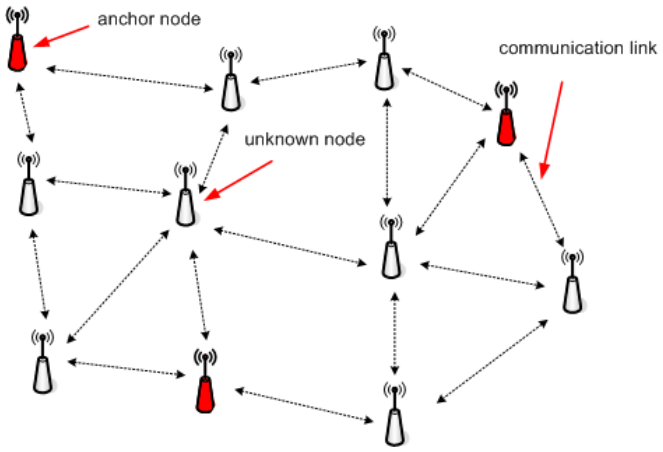
Overview of cooperative localization techniques

2.1 Introduction

Wireless sensor network (WSN) localization is an important task, in which the goal is to obtain estimates of each sensor's position as well as accurately representing their uncertainties. Equipping every sensor with a GPS receiver may be expensive, energy prohibitive and limited to outdoor applications [75]. Therefore, we consider the problem in which some small number of sensors, called *anchor nodes*, obtain their coordinates via GPS or by installing them at points with known coordinates, and the rest, *unknown nodes*, must determine their own coordinates using the anchor nodes and measured inter-sensor distances. If unknown nodes were capable of high-power transmission, they would be able to make measurements with all anchor nodes. This represents *single-hop* localization (Figure 2.1a). However, we prefer to use energy-conserving devices without energy necessary for long-range communication. In this case, each unknown node has available only the noisy measurements of the distance to several neighboring nodes (not necessarily anchor nodes). In other words, we still allow unknown nodes to make measurements with anchor nodes (if possible), but now we additionally allow unknown nodes to make measurements with other unknown nodes. It is still necessary that there is minimum of three (for 2D) or four (for 3D) anchor nodes in the network, but not necessarily directly connected to all unknown nodes. This technique, known as *multi-hop* (or *cooperative*) localization (Figure 2.1b), is the main topic of this thesis.



(a)



(b)

Figure 2.1: (a) Single-hop and (b) multi-hop (cooperative) localization.

2.1.1 Motivating applications

We review few important applications of cooperative localization in WSN. In environmental monitoring applications (such as bush fire surveillance, or water quality monitoring), the measurement data are meaningless without knowing the location from where the data are obtained. For example, it is extremely important to know the location of the sensor which detects the high temperature. For the biological research, it is also very useful to know location of the animals over time. Using multihop routing of the data through the network enables low transmit powers from the animal tags. Furthermore, inter-animal distances, which are of particular research interest, can be estimated using pairwise measurements and cooperative localization methods (without resorting to GPS). The main result of the longer battery lifetimes

is less frequent recollaring of the animals. As another example, we can consider deploying a sensor network in a manufacturing floor. The monitoring and control of equipment has traditionally been wired, but making them wireless reduces the high cost of cabling and makes the manufacturing floor more dynamic. In addition, these sensors monitor storage conditions (temperature and humidity) and help control the heating, ventilation, and air conditioning system. Sensors on mobile equipment report their location when the equipment is lost or needs to be found, and can even contact security if the equipment is about to leave the building. Moreover, location estimation may enable many of applications such as search-and-rescue, intrusion detection, road traffic monitoring, health monitoring, reconnaissance, and surveillance. A description of number of interesting applications can be found in [21, 34, 76, 98].

2.1.2 Classification of cooperative localization methods

Range-based vs range-free methods

Range-free or *connectivity-based* localization methods [8, 68, 100, 101, 109] rely on connectivity between the nodes. The principle of these algorithms is to determine whether or not a sensor is in the transmission range of another sensor. The most attractive feature of the range-free algorithms is their simplicity. However, they can only provide a coarse grained estimate of each node's location, which means that they are not only suitable for applications requiring precise location estimate. *Range-based* or *distance-based* localization algorithms [46,68,78,82,97] use the inter-sensor distance measurements in a sensor network to locate the entire network. This type of algorithms is usually more accurate, but sensitive to measurement errors.

Centralized vs distributed methods

Based on the approach of processing the individual inter-sensor data, localization algorithms can be also considered in two main classes: *centralized* and *distributed* algorithms. Centralized algorithms [97, 100] utilize a single central processor (i.e., fusion center) to collect all the individual inter-sensor data and produce a map of the entire sensor network, while distributed algorithms [46, 68, 82, 97, 118] rely on self-localization of each node in the sensor network using the local information it collects from its neighbors. From the perspective of location estimation accuracy, centralized algorithms are likely to provide more accurate location estimates than distributed algorithms. However, centralized algorithms suffer from the scalability problem, and generally are not feasible to be implemented for large scale WSN. On the other hand, the main disadvantage of the distributed methods is that they require multiple

iterations to converge, which may cause the localization process to take long time. From the communication energy consumption perspective, centralized algorithms in large-scale networks require each sensor's measurements to be sent over multiple hops to the fusion center, while distributed algorithms require only local information exchange between neighboring nodes but many such local exchanges may be required (depending on the number of iterations needed for convergence). If in a given sensor network and distributed algorithm, the average number of hops to the fusion center exceeds the necessary number of iterations, then the distributed algorithm will be more energy-efficient than a typical centralized algorithm [76].

Anchor-based vs anchor-free methods

Anchor-based [8, 68, 78, 97, 109] methods assume that a certain minimum number of the nodes know their position, e.g., by manual placement or using some other location mechanism such as GPS. This localization method has the limitation that it needs another localization system to find the anchor node positions. In contrast, *anchor-free* [46, 82, 100, 101] algorithms use local distance information to attempt to determine node coordinates when no nodes have pre-defined positions. Of course, any such coordinate system will not be unique and can be embedded into another global coordinate space in infinitely many ways, depending on global translation, rotation, and flipping. Therefore, the main problem with anchor-free methods is the need for an additional algorithm for transformation from the relative to the absolute coordinates.

Probabilistic vs deterministic methods

Deterministic algorithms [68, 82, 97, 100, 101, 109] use the measurements to estimate the point estimate of the positions by applying classical least squares, multidimensional scaling, multilateration, or other optimization methods. In favor of their relative computational simplicity, they often lack a statistical interpretation, and as one consequence typically do not provide an estimate of the remaining uncertainty in each sensor location. However, iterative least-squares methods, like N-hop multilateration [97], have a straightforward statistical interpretation, by assuming a Gaussian model for all uncertainties, which may be questionable in practice. Non-Gaussian uncertainty is a common occurrence in real-world sensor localization problems, where there is usually some fraction of highly erroneous (outlier) measurements. On the other hand, *probabilistic* (or *Bayesian*) methods [8, 46, 48, 78, 118] take into account uncertainty of the measurements, so given the likelihood of e.g., measured distance and a prior PDF of the positions of all unknown nodes, they

estimate the posterior PDF of the positions of all unknown nodes. However, the main drawback of the probabilistic methods is the high computational and communication cost which, in some applications, makes these methods unacceptable in low-power WSN. Nevertheless, the particle-based approximation via nonparametric representation, and an appropriate factorization of the PDFs, make probabilistic methods acceptable for inference in sensor networks. In addition, nonparametric representation enables us to estimate any PDF that does not exist in analytical (parametric) form.

2.2 Measurement techniques

Measurement techniques for WSN localization can be broadly classified into three categories:

- Distance-based measurements (RSS, TOA, and TDOA)
- Angle-of-arrival (AOA) techniques
- RSS profiling techniques (fingerprinting)

We describe all of them in this section, with emphasis on the most common used, distance related techniques. The detailed description can also be found in [43, 63, 76, 98].

2.2.1 Received signal strength (RSS)

The goal of this technique is to estimate distance between neighboring sensors from the RSS measurements. These techniques are based on a standard feature found in most wireless devices, a RSS indicator. They are attractive because they require no additional hardware, and are unlikely to significantly impact local power consumption, sensor size and thus cost.

Let us denote this received power by $P_r(d)$. This power varies as the inverse square of the distance d between transmitter and receiver through the Friis equation: [84]:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2} \tag{2.1}$$

where P_t is the transmitted power, G_t is the transmitter antenna gain, G_r is the receiver antenna gain, and λ is the wavelength of the transmitted signal in meters. However, the free-space model is an over-idealization, and the propagation of a signal is affected by reflection, diffraction and scattering. Of course, these effects are

environment (indoors, outdoors, rain, buildings, etc.) dependent. It is accepted on the basis of empirical evidence to model RSS ($P_r(d)$) as a random and log-normally distributed random variable with a distance dependent mean value:

$$P_r(d)[dBm] = P_0(d_0)[dBm] - 10n_p \log_{10}(\frac{d}{d_0}) + X_\sigma \tag{2.2}$$

where $P_0(d_0)$ is known reference power value in dB milliwatts at a reference distance from the transmitter, n_p is the path loss exponent that measures the rate at which the RSS decreases with distance, typically between two and four depending on the specific propagation environment, X_σ is a zero mean Gaussian distributed random variable with standard deviation σ and it accounts for the random effects of shadowing. It is trivial to conclude from (2.2) that, given $P_r(d)[dBm]$, the estimated distance between a transmitter and receiver is:

$$d = d_0 \cdot 10^{-\frac{P_r(d)[dBm] - P_0(d_0)[dBm]}{10n_p}} \cdot 10^{\frac{X_\sigma}{10n_p}} \tag{2.3}$$

As we can see, the distance error is multiplicative (i.e., log-normally distributed) which means that RSS-based distance estimates have variance proportional to their true distance. Therefore, RSS is most valuable in high-density sensor networks.

However, in addition to the path loss, measured RSS is also a function of the calibration of both the transmitter and receiver. Depending on the expense of the manufacturing process, RSS indicator circuits and transmit powers will vary from device to device. Also, transmit powers can change as batteries deplete. All these problems make RSS-based methods suitable only for coarse-grained localization.

2.2.2 Time of arrival (TOA)

Distances between neighboring sensors can be estimated from the propagation time measurements between transmitter and receiver, using two types of measurements, one-way and round-trip.

One-way propagation time measurements measure the difference between the sending time of a signal at the transmitter and the receiving time of the signal at the receiver. It requires the local time at the transmitter and the local time at the receiver to be accurately synchronized. This requirement may add to the cost of sensors by demanding a highly accurate clock and/or increase the complexity of the sensor network by demanding a sophisticated synchronization mechanism. This disadvantage makes one-way propagation TOA measurements a less attractive option than measuring round-trip time in WSNs.

Round-trip propagation TOA measurements measure the difference between the

time when a signal is sent by a sensor and the time when the signal returned by a second sensor is received at the original sensor. Since the same clock is used to compute the round-trip propagation time, there is no synchronization problem. The major error source in round-trip propagation TOA measurements is the delay required for handling the signal in the second sensor. This internal delay is either known via a priori calibration, or measured and sent to the first sensor to be subtracted.

A recent trend in propagation time measurements is the use of ultra-wide band (UWB) signals for accurate distance estimation [40, 102]. UWB is a signal whose fractional bandwidth (the ratio of its bandwidth to its center frequency) is larger than 0.2 or a signal with a total bandwidth of more than 500 MHz. UWB can achieve higher accuracy because its bandwidth is very large and therefore its pulse has a very short duration. This feature makes possible fine time resolution of UWB signals and easy separation of multipath signals.

Generally, errors in TOA estimation are caused by two problems:

- *Early-arriving multipath*: Many multipath signals arrive very soon after the line-of-sight (LOS) signal, and their contributions to the cross-correlation obscure the location of the peak from the LOS signal.

- *Attenuated LOS*: The LOS signal can be severely attenuated compared to the late-arriving multipath components, causing it to be "lost in the noise" and missed completely; this leads to large positive errors in the TOA estimate.

2.2.3 Time difference of arrival (TDOA)

Taking time differences of TOA measurements eliminates the clock bias nuisance parameter. This measurement is done between one transmitter and a number of receivers. The TDOA between a pair of receivers i and j is given by:

$$\Delta t_{ij} = t_i - t_j = \frac{1}{c}(\|r_i - r_t\| - \|r_j - r_t\|) \tag{2.4}$$

where t_i and t_j are the time when a signal is received at receivers, r_i, r_j and r_t is locations of transmitter, c is the propagation speed of the signal. However, the most widely used method is the generalized cross-correlation method [54], where the cross-correlation function between two signals s_i and s_i received at the receivers is given by:

$$\rho_{ij}(\tau) = \frac{1}{T}\int_0^T s_i(t)s_j(t-\tau)dt \tag{2.5}$$

The cross-correlation function can also be obtained from an inverse Fourier transform of the estimated frequency domain cross-spectral density function. Frequency domain processing is often preferred because the signals can be filtered prior to computation of the cross-correlation function. The cross-correlation approach requires very accurate synchronization among receivers but does not impose any requirement on the signal transmitted by the transmitter.

This measurement technique is able to achieve better accuracy than RSS and TOA. However, the accuracy is achieved at the expense of higher equipment cost. The accuracy of TDOA measurements will improve when the separation between receivers increases because this increases differences between time-of-arrival. Closely spaced multiple receivers may give rise to multiple received signals that cannot be separated. Another factor affecting the accuracy of TDOA measurements is multipath. Overlapping cross-correlation peaks due to multipath usually cannot be resolved.

2.2.4 Lighthouse approach

Another interesting approach to distance measurements is the lighthouse approach [85] which derives the distance between an optical receiver and a transmitter of a parallel rotating optical beam by measuring the time duration that the receiver retains in the beam. Figure 2.2 illustrates the principle of the lighthouse approach.

A transmitter located at the origin is equipped with an optical beam whose beam width b is constant with respect to the distance from the rotational axis of the beam. The optical beam rotates at an unknown angular velocity ω around the Z axis. An optical receiver in the XY plane and at a distance d_1 from the Z axis detects the beam for a time duration t_1. From Figure 2.2, it can be shown that:

$$d_1 \approx \frac{b}{2\sin(\alpha_1/2)} = \frac{b}{2\sin(\omega t_1/2)} \tag{2.6}$$

The unknown angular velocity ω can be derived from the difference between the time instant when the optical receiver first detects the beam and the time instant when the optical receiver detects the beam for the second time. Therefore the distance d_1 can be derived from the time duration t_1 that the optical receiver retains in the beam.

A major advantage of the lighthouse approach is the optical receiver can be of a very small size. However, the transmitter may be large and this approach requires a direct LOS between the optical receiver and the transmitter.

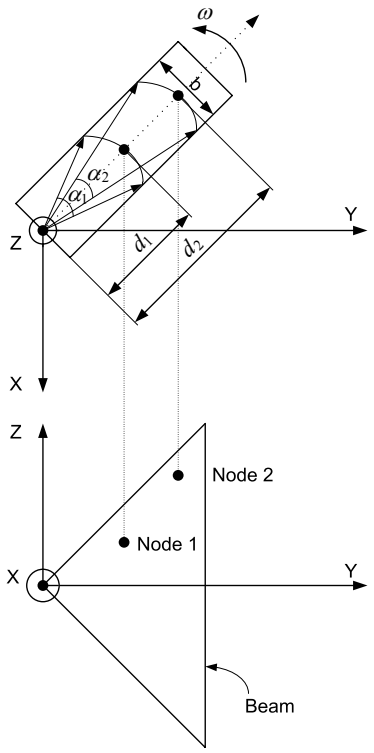


Figure 2.2: Illustration of lighthouse approach

2.2.5 Angle of arrival (AOA)

By providing information about the direction to neighboring sensors rather than the distance to neighboring sensors, AOA measurements [69] provide localization information complementary to the TOA and RSS measurements discussed above. AOA can be divided into two subclasses: those making use of the receiver antenna's amplitude response and those making use of the receiver antenna's phase response.

Beamforming

The basis of the first category is beamforming, using of anisotropy in the reception pattern of an antenna (Figure 2.3). The beam of the receiver antenna is rotated electronically or mechanically, and the direction corresponding to the maximum signal strength is taken as the direction of the transmitter. Relevant parameters are the sensitivity of the receiver and the beam width.

The receiver cannot differentiate the RSS variation due to the varying amplitude of the transmitted signal and the signal strength variation caused by the anisotropy in the reception pattern. One approach to dealing with the problem is to use a second non-rotating and omnidirectional antenna at the receiver. By normalizing the RSS

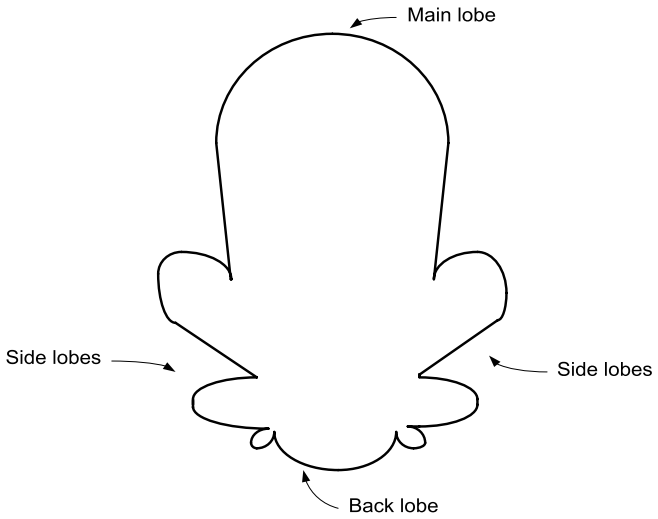


Figure 2.3: Typical anisotropic antenna

received by the rotating anisotropic antenna with respect to the RSS received by the non-rotating omnidirectional antenna, the impact of varying RSS can be largely removed. Another widely used approach is to use a minimum of two (but typically 4) stationary antennas with known, anisotropic antenna patterns. Overlapping of these patterns and comparing the RSS received from each antenna at the same time yields the transmitter direction, even when the RSS changes. Coarse tuning is performed by measuring which antenna has the strongest signal, and it is followed by fine tuning which compares amplitude responses. Because small errors in measuring the RSS can lead to a large AOA measurement error, a typical measurement accuracy for four antennas is 10-15 degrees. With six antennas, this can be improved to about 5 degrees, and 2 degrees with eight antennas [63].

Phase interferometry

The second category of measurement techniques, known as phase interferometry, derives the AOA measurements from the measurements of the phase differences in the arrival of a wave front. It typically requires a large receiver antenna (relative to the wavelength of the transmitter signal) or an antenna array. Figure 2.4 shows an antenna array of N antenna elements.

The adjacent antenna elements are separated by a uniform distance d. The distance between a transmitter far away from the antenna array and the i^{th} antenna element can be approximated by:

$$R_i \approx R_0 - id\cos\theta \tag{2.7}$$

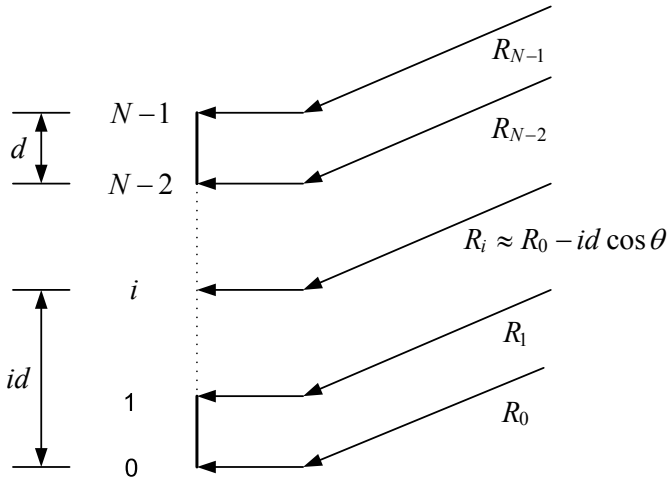


Figure 2.4: An antenna array with N antenna elements

where R_0 is the distance between transmitter and the 0^{th} antenna and θ is the bearing of the transmitter with respect to the antenna array. The transmitter signals received by adjacent antenna elements have a phase difference $2\pi d\cos\theta/\lambda$, which allows us to obtain the bearing of the transmitter from the measurements of the phase difference. This approach works quite well for high signal-to-noise ratio but may fail in the presence of strong co-channel interference and/or multipath signals.

The accuracy of AOA measurements is limited by the directivity of the antenna, by shadowing and by multipath reflections. AOA measurements rely on a direct line-of-sight path from the transmitter to the receiver. However a multipath component may appear as a signal arriving from an entirely different direction and can lead to very large errors. Multipath problems in these measurements can be addressed by using the maximum-likelihood (ML) algorithms [63, 83]. Typically ML methods will estimate the AOA of each separate path in a multipath environment. The implementation of these methods is computationally very intensive and requires complex multidimensional search. Another class of ML methods assumes that the structure of the signal waveform is known at the receiver. This extra information improves the accuracy of AOA measurements and simplify computation. However, due to the high equipment cost, AOA methods are rarely used for WSN localization.

2.2.6 RSS profiling technique

RSS profiling-based (fingerprinting) technique [67,124], works by constructing a form of map of the signal strength in the coverage area. The map is obtained either offline by a priori measurements or online using sniffing devices [60] deployed at known

locations. They have been mainly used for location estimation in wireless local area networks (WLAN), but they would appear to be attractive also for WSN. In this technique, in addition to anchor nodes (e.g. access points in WLANs) and non-anchor nodes, a large number of sample points (e.g. sniffing devices) are distributed throughout the coverage area of the WSN. At each sample point, a vector of RSS is obtained, with the i^{th} entry corresponding to the i^{th} anchor's transmitted signal. Of course, many entries of the signal strength vector may be zero or very small, corresponding to anchor nodes at larger distances (relative to the transmission range or sensing radius) from the sample point. The collection of all these vectors provides (by extrapolation in the vicinity of the sample points) a map of the whole region. The collection constitutes the RSS model, and it is unique with respect to the anchor locations and the environment. The model is stored in a central location. By referring to the RSS model, a non-anchor node can estimate its location using the RSS measurements from anchors. However, the main problem of this approach is sensitivity to environmental changes. In that case, the system must be re-calibrated, i.e., new vector of RSS have to be collected.

2.3 Deterministic localization techniques

In this section, we review deterministic (or non-Bayesian) localization techniques, in which the main goal is to find the point estimate of the sensor positions. In particular, we focus on connectivity-based and distance-based cooperative localization algorithms due to their prevalence in cooperative WSN localization. For both classes, we describe few centralized and distributed algorithms. In addition, we describe a method for localization using AOA measurements.

2.3.1 Connectivity based algorithms

Connectivity-based or "range-free" localization algorithms do not rely on any of the measurement techniques described in Section 2.2. Instead they use the connectivity information to estimate the location of the unknown nodes (i.e., who is within the communication range). We will describe three algorithms in subsequent sections: distributed ad-hoc positioning based on the *distance-vector-hop* (DV-hop) approach [68], centralized and distributed algorithm based on *multi-dimensional scaling* (MDS) [100, 101], and distributed *concentric anchor beacon* (CAB) [109].

Ad-hoc positioning

This method extends the capabilities of GPS to non-GPS network in hop by hop fashion in an ad-hoc network [68]. Positioning is based on hybrid method combining approximation of distance vector and GPS triangulation. For the anchor nodes in the network it is assumed to be placed at random position because there are a lot of applications with inaccessible deployment area where the anchors are usually scattered from the air. In this case, one option is to use hop by hop propagation capability of the network to forward messages (hop-count) to anchors. Once an arbitrary node has estimates to a number of minimum 3 anchors, it can compute its own position using a similar procedure with the one used in GPS [75].

First phase of the algorithm is DV-hop propagation, a classical distance vector exchange. Each node maintains a table with coordinates and hop-counts $\{x_i, y_i, h_i\}$, and exchange updates only with its neighbors. Once an anchor gets distances to other anchors, it estimates a average distance between two neighbors, which is then deployed as a correction to the entire network. When receiving the correction, an arbitrary node may then have estimate distances to anchors, in meters, which can be used to perform the triangulation. The correction that anchor $\{x_i, y_i\}$ computes is:

$$c_i = \frac{\sum \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\sum h_i}, \quad \text{(for all anchors } j, i \neq j) \tag{2.8}$$

In the example in Figure 2.5, nodes A1, A2 and A3 are anchors, so node A1 has both the Euclidean distance to A2 and A3, and the path length of 2 hops and 6 hops, respectively. A1 then computes the correction $(80 + 30)/(6 + 2) = 13.75$, which is in the fact the estimated average distance between neighbors. A1 has then choice of either computing a single correction to be broadcasted into the network, or preferentially send different corrections along different directions. In a similar manner, A2 computes correction of $(30 + 60)/(2 + 5) = 12.86$ and A3 a correction of $(60 + 80)/(6 + 5) = 12.73$.

Unknown node gets an update from one of the anchors, and it is usually the closest one, depending on the deployment policy. Corrections are distributed by controlled flooding, meaning that once a node gets and forwards a correction, it will drop all the subsequent ones. This policy ensures that most nodes will receive only one correction, from the closest anchor. Controlled flooding helps keeping the corrections localized in the neighborhood of the anchors they were generated from, thus accounting for nonisotropies across the network. In the above example, assume U gets an correction from anchor A2, so its estimated distances to the three anchors will be: to A1: 3×12.86, to A2: 2×12.86, and to A3: 3×12.86. These values

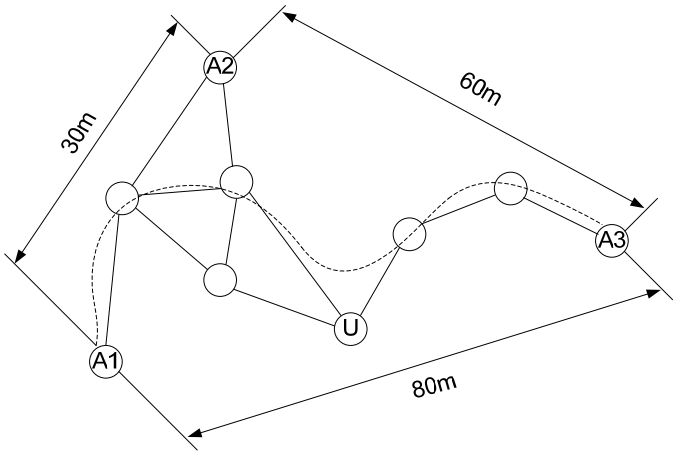


Figure 2.5: DV-hop correction example

are then plugged into the triangulation procedure, second phase of this algorithm, to get a location estimate of node U.

The second phase of the algorithm is triangulation similar to GPS triangulation. GPS triangulation [75] uses at least four satellites and the clock bias of the receiver. In this case, we are only dealing with distances, so there is no need for clock synchronization. Moreover, in this 2D case, we need minimum three anchor nodes (equivalent to satellites in GPS). This problem can be also solved using standard least square method [98].

The advantages of the DV-hop propagation scheme are its simplicity and the fact that it does not depend on measurement error. The drawbacks are that it will only work for isotropic networks, that is, when the properties of the graph are the same in all directions, so that the corrections that are deployed reasonably estimate the distances between hops. Moreover, ad-hoc positioning has the following properties: it is distributed, does not require special infrastructure or setup, provides global coordinates and requires recomputation only for moving nodes.

Multi-dimensional scaling (MDS)

MDS [12] is an efficient technique for the analysis of dissimilarity of data that takes full advantage of connectivity information between nodes. The goal of MDS is to find a low-dimensional representation of a group of objects (e.g., sensor positions), such that the distances between objects fit as well as possible a given set of measured pairwise "dissimilarities" (e.g., inter-sensor distances or hop-counts). There are number of applications of MDS in chemical modeling, economics, sociology, etc. Recently, this method has been also applied for cooperative localization [100,101]. The centralized version of the algorithm (MDS-MAP), builds a global map using classical

Algorithm 1 Classical MDS

1: Compute the squared distance matrix D^2, where $D = [d_{ij}]_{n \times n}$
2: Compute the centering operator: $J = I - ee^T/n$, where $e = (1, 1, ..., 1)^T$
3: Apply double-centering to D^2: $H = -\frac{1}{2} J D^2 J$
4: Compute the singular-value decomposition (SVD) of matrix H: $H = UVU^T$
5: For i dimensional map, create sub-matrices V_i and U_i, which include i largest eigenvalues and their corresponding eigenvectors.
6: Compute the coordinates: $X = U_i V_i^{1/2}$

metric MDS. Classical metric MDS is the simplest case of MDS: The data is quantitative and the proximities of objects are treated as distances in a Euclidean space. The goal is to find a configuration of points in a multidimensional space (2D or 3D, in case of localization) such that the interpoint distances are related to the provided proximities by some transformation (e.g., a linear transformation). If the proximity data were measured without error in a Euclidean space, then classical metric MDS would exactly recreate the configuration of points. Because classical metric MDS has an analytical solution, it can be performed efficiently on large matrices.

MDS-MAP method consists in three steps:

- Compute the shortest paths between all pairs of nodes in the region of consideration. The shortest path distances are used to construct the distance matrix for MDS.

- Apply MDS to the distance matrix, retaining the first two largest eigenvalues and eigenvectors to construct a 2D relative map (see Alg. 1).

- Given sufficient anchor nodes (three or more for 2D), transform the relative map to an absolute map based on the absolute positions of anchors.

In the first step, it is necessary to assign distances to the edges in the connectivity graph. When we only have connectivity information, a simple approximation is to assign value 1 to all edges. Then, compute shortest-path for all pairs of the nodes. The time complexity is $O(n^3)$, where n is the number of nodes. In the second step, classical MDS is applied directly to the distance matrix. The result of MDS is a relative map that gives a location for each node. Although these locations may be accurate relative to one another, the entire map will be arbitrarily rotated and flipped relative to the true node positions. In the last phase, the relative map is transformed through a linear transformation, which may include scaling, rotation, and reflection. The goal is to minimize the sum of the squares of the errors between the true positions of the anchors and their transformed positions in the MDS map.

Computing the transformation parameters takes $O(m^3)$ time, where m is the number of anchors.

MDP-MAP does not work well on irregular networks, because it relies on shortest-path distance estimation, which can have large errors for remote nodes. Another problem with this centralized method is that it is not applied easily to large networks for which reading out the connectivity and distance information is potentially prohibitive. The improved version of MDS-MAP, called MDS-MAP-P, addresses both of these problems.

MDS-MAP-P builds many local maps and then patches them together to form a global map. This method relies on local information and avoids using the distance estimation between remote notes, so it achieves better results on irregular networks. Individual nodes simultaneously compute their own local maps using their local information. Then, these maps can be incrementally merged to form a global map. Therefore, another benefit is that this algorithm can be easily executed in a distributed fashion. MDS-MAP-P method consists in four steps:

- Set the range for local maps, R_{lm}. For each node, neighbors within R_{lm} hops are involved in building its local map.

- For each node, apply MDS-MAP to the nodes within range R_{lm} to generate its local map.

- Merge local maps [100, 101].

- Given sufficient anchor nodes (three or more for 2D), transform the relative map to an absolute map based on the absolute positions of anchors.

The strength of both approaches is that it can be used when there are few or no anchor nodes. This approach also does not have limitation about anchor node placement. It builds a relative map of the nodes even when no anchor nodes are available. With three or more anchor nodes, the relative map can be transformed into absolute coordinates. An optional refinement step can be used to further improve the quality of the solution, at the expense of additional computation. This variant of MDS-MAP (MDS-MAP-PR) [101] requires measured distances between the neighboring nodes (see Section 2.3.2). A patching-based variation not only allows distributed and parallel computation, but also gives better solutions, especially on irregularly-shaped networks.

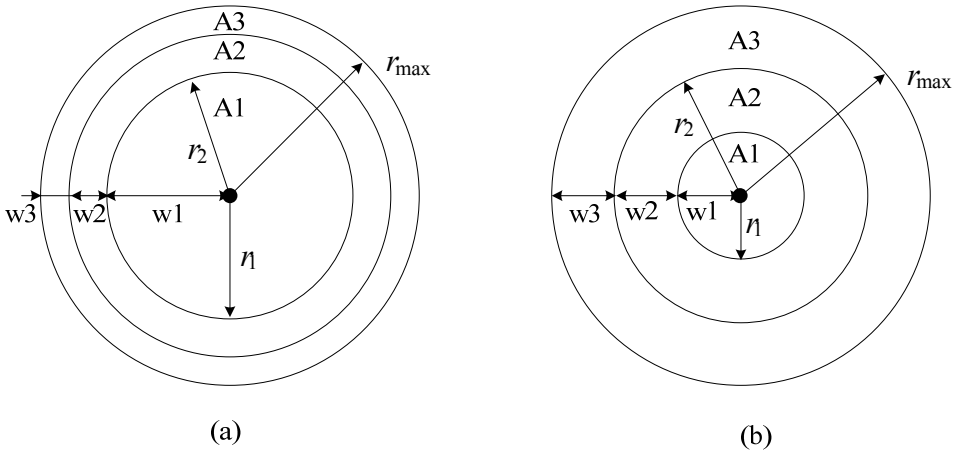


Figure 2.6: Anchor beacon transmission ranges for (a) CAB-EA, and (b) CAB-EW

Concentric anchor beacon (CAB)

CAB [109] localization algorithm is a distributed range-free approach which can use a small number of anchor nodes. Each anchor emits beacons (signal) at different power levels which carries information including the anchor's position, its power level, and the estimated maximum distance that the beacon can travel. From the information received by each beacon heard, nodes can determine in which annular ring they are located within each anchor. Each unknown node uses the approximated center of intersection of the rings as its position estimate.

In a wireless propagation environment, given the signal power transmitted by an anchor node to be P_{tx} the path loss model can determine the average signal power received by an unknown node P_{rcv}. In this case, it is assumed the use of the following path loss model:

$$P_{rcv} = \frac{k \cdot P_{tx}}{r^n} \tag{2.9}$$

where k is a constant, r denotes the distance between the anchor and the unknown node, and n denotes the path loss exponent. Let $P_{threshold}$ denote the minimum required received signal power to decode the beacon signal correctly. It depends on the target bit error rate and the modulation scheme being used. Using this value and (2.9), we can calculate the maximum range $r_{\max}$ between anchor and unknown node such that the sensor can decode the signal correctly:

$$r_{\max} = \left(\frac{k \cdot P_{\max}}{P_{threshold}} \right)^{1/n} \tag{2.10}$$

The proposed CAB algorithm differs from other range-free localization approaches in that anchors transmit several beacon signals at different power levels. This re-

quirement is feasible in current WSNs. Ideally, the different power levels divide the possible transmission ranges of an anchor into a circle and rings. The lowest power level creates a circular coverage area, and the following higher levels are distinguished by rings emanating from this lowest level. There are two variations of this algorithm (Figure 2.6). In CAB-EA, it is assumed that the area of the innermost circle and the rings are all the same; and in CAB-EW, it is assumed that the width of the innermost circle and the rings are all the same.

The relationship between the i^{th} transmitting beacon power level P_i and the maximum transmitting power level $P_{\max}$ is calculated using (2.9) and the relationship between the beacon transmission ranges r_i and the maximum transmission range $r_{\max}$ (according to Figure 2.6). For CAB-EA and CAB-EW, these powers are respectively given by:

$$P_i = \left(\frac{i}{m}\right)^{\frac{n}{2}} P_{\max}, \quad P_i = \left(\frac{i}{m}\right)^{n} P_{\max} \tag{2.11}$$

The CAB localization algorithm (applicable to both CAB-EA and CAB-EW versions) consists in 3 steps:

- Each anchor transmits the beacon signals at varying power levels consecutively, which includes the anchor's ID, the anchor's location, the transmitting power level P_i, and the estimated maximum distance that the beacon signal can be heard.
- Each unknown node listens for beacons and collects the anchor's information and determines within which region of the anchor's concentric transmission circles it lies (Figure 2.7).
- The final position estimate is computed as the average of all the valid intersection points.

Depending on the percentage of anchors deployed, each unknown node can hear multiple beacons from different anchors. For computational simplicity, information from at most three neighboring anchors is used to estimate a sensor's location. The result is also valid when the unknown node only receives beacon signals from two neighboring anchors. On the other hand, if the unknown node receives beacon signals from only one anchor, either a random coordinate within the ring that the unknown node resides will be chosen as the position estimate, or the error should be reported.

There are three important advantages of the CAB localization algorithm. First, CAB is distributed and simple to implement. For the anchors, their only task is to transmit beacon signals with different power levels. For each unknown node,

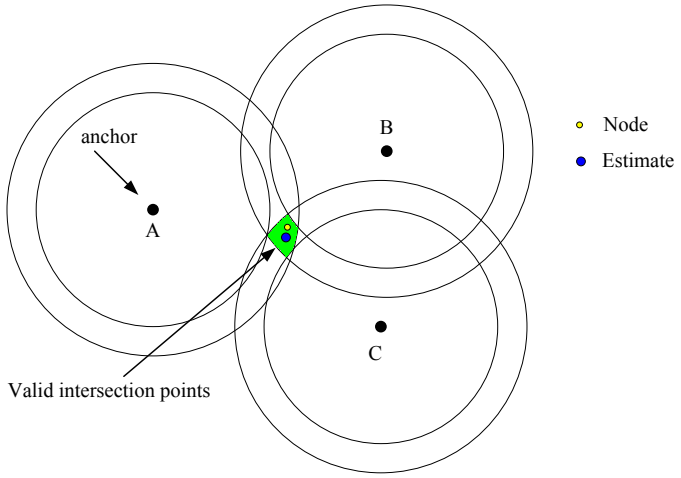


Figure 2.7: Example of localization using CAB

the determination of the intersection points from three chosen anchors as well as the position estimate by averaging are not computationally intensive. Second, no information exchange between neighboring sensors is necessary. This reduces the energy requirement for localization. And third, maintain high accuracy comparing to other connectivity-based algorithms. On the other hand, a sensor has to be able to transmit several beacon signals at sufficient number of power levels, what makes this method expensive.

2.3.2 Distance based algorithms

The core of distance based localization algorithms is the use of inter-sensor distance measurements in a WSN to locate the entire network. The main measurements techniques used for this approach are described in Section 2.2.1. We first describe the extension of connectivity based algorithms (Section 2.3.1) to make them applicable for distance based methods. Then, we will provide the detailed description of two well-known algorithms: *collaborative (N-hop) multilateration* (distributed and centralized version) [97], and *anchor-free* distributed (AFL) localization [82].

Extension of connectivity-based algorithms

The centralized MDS-MAP approach [100,101], used in the connectivity-based localization algorithms described in Section 2.3.1, can be readily extended to incorporate distance measurements into the corresponding optimization problem. Only a refinement step has to be added between steps 2 and 3. In this step, using the position estimates of nodes in the MDS solution as an initial solution, least-squares minimization can be applied to improve the match between the measured distances

between neighboring nodes and their distances in the solution. However, the main problem of this approach is that we cannot easily and accurately estimate n-hop distances (n=2,3...). Alternative approach [50] tries to estimate these distances using iterative MDS, in which random initial configuration of the nodes is used for the computation of unavailable distances. Another method, distributed weighted MDS (DW-MDS) [23], uses weights to quantify the accuracy of the measurements between each pair of the nodes. Hence, if the measurement is not available, the weight is set to zero.

Distributed ad-hoc DV-hop algorithm, described in Section 2.3.1, can be extended to incorporate distance measurements. A modified version of this algorithm, called DV-distance [68], includes distance measurements into the localization process. The only difference is propagation of measured distance among neighboring nodes instead of hop-count.

Collaborative (N-hop) multilateration

The collaborative multilateration [97] algorithm will be presented in two computation models, centralized and distributed. In centralized algorithm all computation takes place at a base station, and in distributed algorithm computation takes place at every node. One of the main challenges in this algorithm is to prevent error accumulation inside the network. To prevent it, the node localization problem is set up as a least squares estimation problem with respect to the global network topology. Collaborative multilateration takes place in three main phases:

- Formation of collaborative subtrees
- Computation of initial estimates
- Position refinement

In the first phase, it is necessary to form the subtrees. Collaborative subtrees constitutes a configuration of unknowns and anchors for which the estimated location can be uniquely determined. The nodes that do not meet the criteria for collaborative subtrees cannot participate in this configuration. The position estimates for such nodes are determined later in a post-processing phase. In the single-hop setup of Figure 2.8(a), the basic requirement for unknown node is that it is within range of at least three anchors which are not lie in a straight line. A two-hop scenario (Figure 2.8(b)) represents the case where the anchors are not always directly connected to the node, but they are within a two-hop radius from the unknown node. In this case, the first condition is the same like for one-hop scenario, but these nodes are

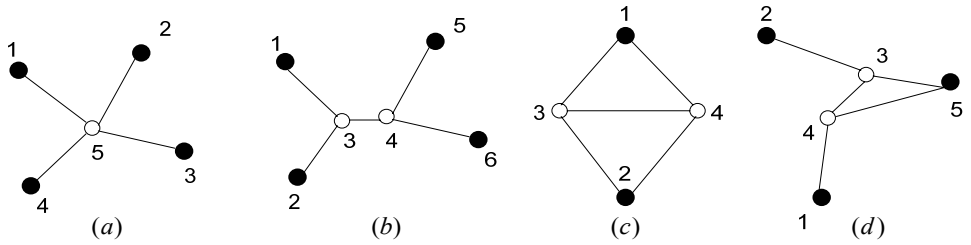


Figure 2.8: (a) One-hop, (b) Two-hop, (c) Two-hop (symmetric case), (d) Two-hop (with independent reference.)

not required to be anchors. The second condition is that an unknown node uses at least one reference point that is not collinear with the rest of its reference points. If all reference points lie in a straight line, then the unknown node will have two possible positions. Another type of problem is symmetrical setup (Figure 2.8(c)), when two nodes can be swapped without any violation of the constraints. Therefore, the third condition is that each pair of unknown nodes that use link to each other as a constraint, has at least one independent reference (Figure 2.8(d)).

N-hop scenario requires similar set of criteria. Starting from an unknown node, we test if it has at least three neighbors with unique positions. If the node has three neighbors that do not already know if their solution is unique, then recursive call is executed at each neighbor to determine if its position is unique. To meet the requirement of third condition of two-hop case, each node used as an independent reference is marked as used. This prevents other nodes from subsequent recursive calls to re-use that node as an independent. For example, in the network in Figure 2.9(a), all nodes satisfy requirements, but in Figure 2.9(b), node 5, which has only two neighbors, can not be a part of the subtree.

The initial estimates are obtained by applying the distance measurements as constraints on the x and y coordinates of the unknown nodes. If the distance between an unknown node and the anchor A is a then the x coordinates of node C, are bounded by a, to the left and to the right of the x coordinate of anchor A, $x_a - a$ and $x_a + a$ (Figure 2.10). Similarly, node C is two hops away from the anchor B, so it is bounded by $x_b - (b + c)$ and $x_b + (b + c)$. By knowing this information, the final bounds for C are $x_a - a$ and $x_b + (b + c)$. This operation, called *min-max*, selects the tightest left hand side bound and the tightest right hand side bound from each anchor. The same operation is done on the y coordinate. The node then combines its bounds to obtain a *bounding box* of the region where the node lies. To obtain this bounding box, the location of all anchors is forwarded to all unknowns along a minimum weight path.

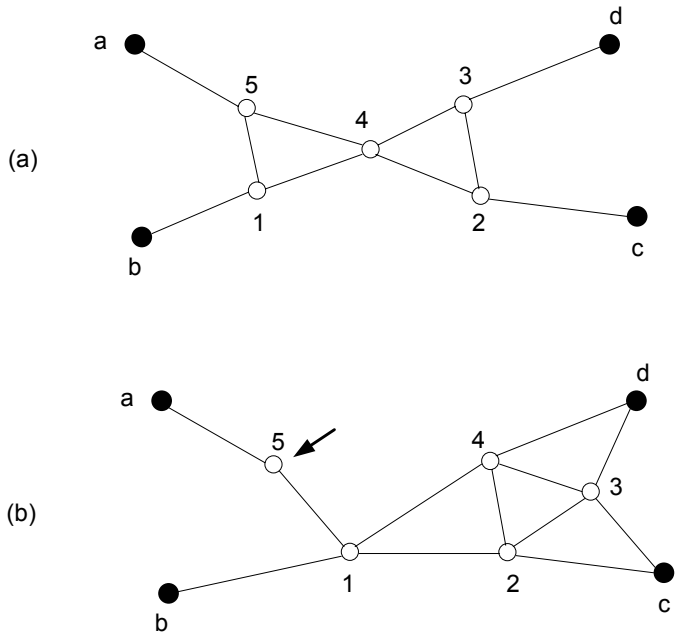


Figure 2.9: N-hop scenario: (a) regular, and (b) irregular case

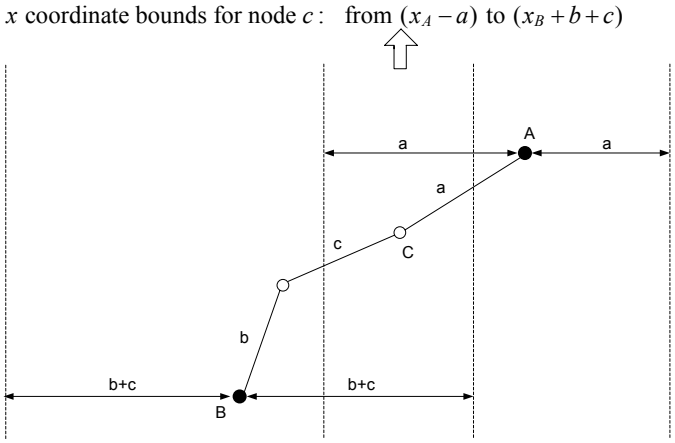


Figure 2.10: Initial estimates for node C

The initial position estimate of a node is taken to be as center of the bounding box. Therefore, for example in Figure 2.10, the initial estimates for the node C are:

$$x_c = \frac{x_a - a + x_b + (b + c)}{2}, \quad y_c = \frac{y_a - a + y_b + (b + c)}{2} \tag{2.12}$$

Third phase, position refinement, can be implemented in two possible computation models, centralized or distributed, so we describe both of them. Using the collaborative subtrees and the initial position estimates, the unknown node position estimates can be computed at a central unit. The objective is to minimize the resid-

uals between the measured distances between the nodes and the distances computed in second phase using equation:

$$f_{ij} = d_{ij} - \sqrt{(ex_i - x_j)^2 + (ey_i - y_j)^2} \tag{2.13}$$

where d_{ij} represents the measured distance between nodes i and j, f_{ij} is residual between measured and estimated quantities, and the prefix e in front of x and y denotes estimated coordinates as opposed to known coordinates. The objective is to minimize the mean square error over all equations:

$$F(x_i, y_i, x_{i+1}, y_{i+1}, ...) = \min \sum f_{ij}^2 \tag{2.14}$$

The solution to this optimization problem can be obtained using some of the standard least squares methods, for example Kalman filter (KF) [116]. A KF consists of two phases, a time update phase and a measurement update phase. For this purpose, the network is assumed to be static, the positions of the nodes do not change in time, so the time update phase is not used. The measurement update phase is given by the next set of equations:

$$K_k = P_k^- H^T \left(H P_k^- H^T + R \right)^{-1} \tag{2.15}$$

$$\hat{x}_k = K_k \left(z_k - \hat{z}_k \right) + \hat{x}_k^- \tag{2.16}$$

$$P_k = \left(I - K_k H \right) P_k^- \tag{2.17}$$

where vector $\hat{x}_k^-$ represents initial estimates, $\hat{x}_k$ represents new estimates after the measurements update phases, P_k^- is the a priori estimate of the error covariance, P_k is the new estimate after the measurement update phase. K_k represents the KF gain and it serves a weight to the residual of the filter. The residual is the difference between the measurements (represented by z_k) and the predicted measurement ($\hat{z}_k = H\hat{x}_k^-$). Vector $\hat{z}_k$ is the distance between the nodes, based on the current position estimate, so matrix H is the Jacobian of $\hat{z}_k$ with respect to the a priori estimates ($\hat{x}_k^-$) of the location. Matrix R is the measurement noise covariance matrix, which contains the known covariance of the distance measurement system (e.g., Gaussian noise). Alg. 2 shows how to estimate the unknown locations.

Each edge in the collaborative subtree contributes one entry in the measurement matrix z_k. In matrix H, the number of unknown nodes determines the number of columns and the number of edges determines the number of rows. The noise covariance matrices (P_k, P_k^-) are square matrices whose size depends on the number of unknown nodes (a priori value P_k^- is set to the identity matrix). The measurement noise matrix R is a square matrix with size determined by number of edges in

Algorithm 2 Collaborative multilateration

1: Set the vector to the initial estimates
2: Evaluate the set of equations (2.15)-(2.17)
3: Evaluate the stopping criterion: $\sqrt{(\hat{x}_k)^2 - \left(\hat{x}_k^-\right)^2} \leq \Delta$, where Δ is some pre-defined tolerance. If the criterion is met then the algorithm terminates.
4: Otherwise, set the prediction $\hat{x}_k^-$ to the new estimate $\hat{x}_k$ and go to step 2.

the collaborative subtree (set to identity matrix multiplied by measurement error). Therefore, small increasing in matrix sizes dramatically increase the amount of computation that has to be performed, so such computation cannot be performed using a low cost microcontroller available on the sensor nodes. This is the reason why we need a distributed approximation where every node participates in the computation.

Finally, the eventual post-processing phase uses the computed node estimates to refine the position estimates of nodes that could not participate in the computation subtree configuration. This phase has the similar functionality as the second phase, but it is more constrained by the newly computed location estimates in the computation subtree.

In the distributed version, of this algorithm, each unknown node is responsible for computing its own location estimate. This is achieved by performing local computation and communication with the neighboring nodes. In this distributed scheme, after completion of the first two phases, each node inside the computation tree computes an estimate of its location. Since most unknown nodes are not directly connected to anchors, they use the initial estimates of their neighbors as the reference points for estimating their location. As soon as an unknown computes a new estimate, it broadcast this estimate to its neighbors, and the neighbors use it to update their own position estimates. The computation is repeated from node to node across the network until all the nodes reach the pre-specified tolerance Δ. In case the process proceeds uncontrolled, then the nodes will converge at local minimum and erroneous estimates will be produced. For example, if two neighboring unknown nodes that compute and broadcast their updates as soon as an update from each other is received, then their updating process will proceed faster that the remaining nodes in the computation subtree. This introduces a *local oscillation* in the computation that makes the nodes converge to their final estimates much faster but without complying with the global gradient. To prevent this problem, the multilateration at each node are executed in a sequence across all the unknown members of the computation subtree. This sequence is repeated until all unknown nodes converge to a pre-specified tolerance.

Anchor-free localization (AFL)

We describe a fully decentralized algorithm, called AFL [82], in which all the nodes start from a random initial coordinate assignment and converge to a consistent solution using only local node interactions. The key idea in AFL is *fold-freedom*, where nodes first configure into a topology that resembles a scaled and unfolded version of the true configuration, and then run a force-based relaxation procedure called *mass-spring based optimization* to correct and balance localized errors. The resulting coordinate assignment has translation and orientation degrees of freedom, but is correctly scaled. A post-process could incorporate absolute position information into three or four anchor nodes to remove the translation and orientation degrees of freedom. We describe both phases of this algorithm.

The goal of the first phase of AFL is to embed the graph structurally similar to the original embedding. More specific, the algorithm tries to avoid folds in the resulting graph compared to the original graph. Thus, it is necessary to define a fold-free embedding of a graph to be one where every cycle of the embedding has the correct clockwise/counterclockwise orientation of nodes with respect to the original graph. It is assumed that each node has a unique identifier; the identifier of node i is denoted by ID_i. The hop-count (h_{ij}) identifies the number of nodes along the shortest radio path between nodes i and j. Assuming symmetrical links between nodes, the graph is undirected ($h_{ij} = h_{ji}$). The algorithm first selects five reference nodes. Four of these nodes n_1, n_2, n_3 and n_4 are selected such that they are on the periphery of the graph and the pair (n_1, n_2) is roughly perpendicular to the pair (n_3, n_4). The node n_5 is elected such that it is in the "middle" of the graph. These five nodes are elected in five steps:

1. Select an arbitrary node n_0. Then, select the reference node n_1 to maximize h_{01} (n_1 is a node that is the maximum hop-count away from node n_0). Any ties are broken using the node's ID.

2. Select reference node n_2 to maximize h_{12}. Again, any ties are broken using the node's ID.

3. Select reference node n_3 to minimize $|h_{13} - h_{23}|$. In general, several nodes may all have the same minimum value, and the tie-breaking rule is to pick the node that maximizes $h_{13} + h_{23}$. This step selects a node that is roughly equidistant from nodes n_1 and n_2, and far away from both of them.

4. As in the previous step, select reference node n_4 to minimize $|h_{14} - h_{24}|$. Now, break ties differently: from among several potential contender nodes, pick the

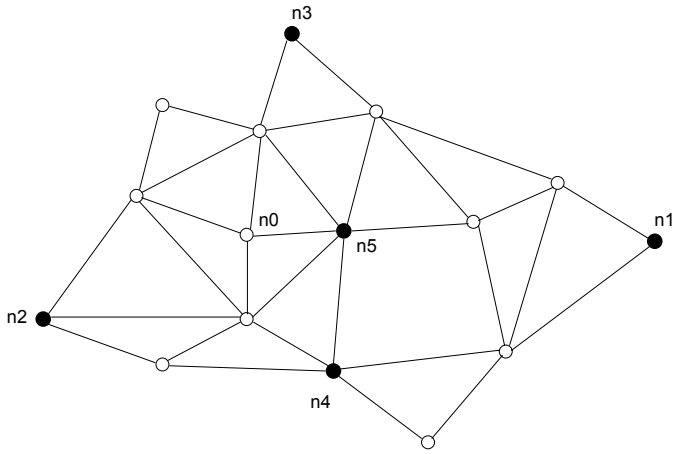


Figure 2.11: The graph obtained after running the fold-free phase

node that maximizes h_{34}. This optimization selects a node roughly equidistant from nodes n_1 and n_2 while being farthest from node n_3.

5. As in the previous step, select reference node n_5 to minimize $|h_{15} - h_{25}|$. From the contender nodes, pick the node that minimizes $|h_{35} - h_{45}|$. This optimization selects the node representing the rough "center" of the graph.

For all other nodes n_i, the hop-counts from the chosen reference nodes (h_{1i}, h_{2i}, h_{3i}, h_{4i} and h_{5i}) and maximum radio range R are used to approximate the polar coordinates (ρ_i, θ_i):

$$\rho_i = h_{5i} \times R,\ \theta_i = \arctan\left(\frac{h_{1i} - h_{2i}}{h_{3i} - h_{4i}}\right) \tag{2.18}$$

This coordinate assignment roughly approximates the true layout of the graph. The use of range to represent one hop-count, results in a graph which is physically larger than the original graph. This property of the graph helps avoid local minima during the optimization phase. One example of this graph is shown in Figure 2.11.

The second phase of the AFL algorithm, the mass-spring optimization, runs concurrently at each node. At any time, each node n_i has a current estimate $\hat{p}_i$ of its position. Each node n_i also periodically sends this position estimate to all its neighbors. Now, each node knows its own estimated position and the estimated position of all its neighbors. Using these position estimates, each node n_i calculates the estimated distance $\hat{d}_{ij}$ to each neighbor n_j. It also knows the measured distance r_{ij} to each neighbor n_j. Let $\hat{v}_{ij}$ represents the unit vector in the direction from $\hat{p}_i$ to $\hat{p}_j$. The force $\vec{F}_{ij}$ in the direction $\hat{v}_{ij}$ and resultant force $\vec{F}_i$ on the node n_i, are

respectively given by:

$$\vec{F}_{ij} = \hat{v}_{ij}(\hat{d}_{ij} - r_{ij}) \,, \quad \vec{F}_i = \sum_j \vec{F}_{ij} \tag{2.19}$$

The energy E_{ij}, due to the difference in the measured and estimated distances, is the square of the magnitude of $\vec{F}_{ij}$, and the total energy of node n_i is given by:

$$E_i = \sum_j E_{ij} = \sum_j (\hat{d}_{ij} - r_{ij})^2 \tag{2.20}$$

so the total energy of the system is $E = \sum_i E_i$.

The energy E_i reduces when the node n_i moves by an infinitesimal amount in the direction of the resultant force $\vec{F}_i$. The exact amount by which each node moves is important for two reasons. First, it must be ensured that the new position has a smaller energy than the original position; second, it must be ensured that such movement does not result in a local minima. AFL can guarantee the first condition by calculating the energy at the new location before moving there to guarantee that the energy reduces. But there is no simple way to guarantee that the move does not result in a local minima. It has been empirically chosen [82] that each node moves by the amount $\left|\vec{F}_i\right| /(2m_i)$, inversely proportional to the number of neighbors of neighbors of m_i. However, thanks to the fold-freedom phase, there is a very low probability of converging to local minima. Even if the graph reaches a local minimum, the fraction of nodes that get displaced tends to be small, thus causing only a small deformation in the resulting graph.

2.3.3 Localization using AOA

We describe a method, called ad-hoc positioning using AOA [69], in which all unknown nodes have to determine their orientation and position in an ad-hoc network where only a fraction of the nodes have positioning capabilities. It is assumed that each node has the AOA capability (see Section 2.2.5). We assume that after the deployment, the axis of the node has an arbitrary unknown heading, represented in Figure 2.12 by a thick black arrow. In this case, the AOA capability provides for each node *bearings* to neighboring nodes with respect to a node's own axis. A *radial* is a reverse bearing, or the angle under which an object is seen from another point.

The term *heading* means the bearing to north, that is, the absolute orientation of the main axis of each node. In Figure 2.12, for node B, bearing to A is $\widehat{ba}$, radial from A is $\widehat{ab}$, and heading is $\hat{b}$. The problem to be solved is: given imprecise bearing measurements to neighbors in a connected ad-hoc network where a small fraction of

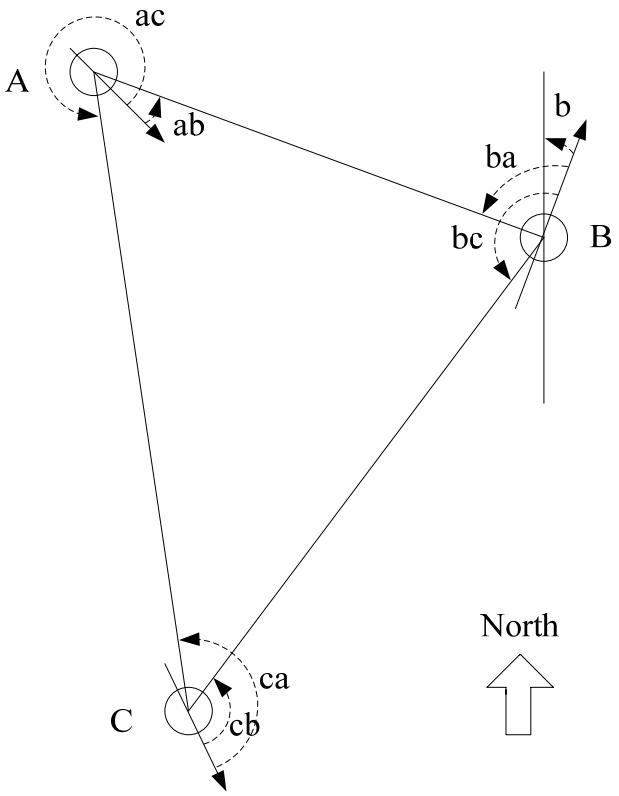


Figure 2.12: Nodes with AOA capability

the nodes have self positioning capability, find headings and positions for all nodes in the network. The difficulty of the problem is the fact that the capable nodes (anchors) comprise only a small fraction of the network, and most regular nodes are the nodes that are not in direct contact with enough anchors.

The original ad-hoc concept has been shown to work using range measurements (see Section 2.3.2), but is in fact extensible to angle measurements. It is a method which can forward orientation so that nodes which are not in direct contact with the anchors can still infer their orientation with respect to the anchor. The term "orientation" means either bearing or radial. We describe two algorithms, DV-Bearing, which allows each node to get a bearing to an anchor, and DV-Radial, which allows a node to get a bearing and a radial to a anchor. The propagation works very much like a mathematical induction proof. The fixed point: nodes immediately adjacent to an anchor get their bearings/radials directly from the anchor. The induction step: assuming that a node has some neighbors with orientation for a anchor, it will be able to compute its own orientation with respect to that anchor, and forward it further into the network. What remains to be found is a method to compute this induction step, both for bearings and radials.

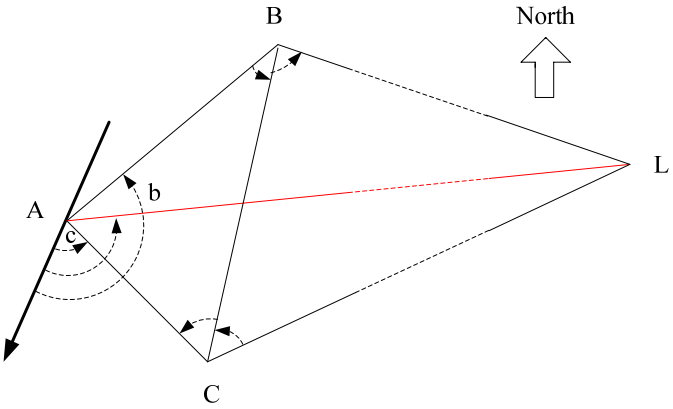


Figure 2.13: Illustration of AOA algorithm: Node A infers its bearing to L using B's and C's bearings to L

The method is shown in Figure 2.13: assume node A knows its bearings to immediate neighbors B and C (angles $\widehat{b}$ and $\widehat{c}$), which in turn know their bearings to a faraway landmark L. The problem is for A to find its bearing to L. If B and C are neighbors of each other, then A has the possibility to find all the angles in triangles ΔABC and ΔBCL. This would allow A to find the angle $\widehat{LAC}$, which yields the bearing of A with respect to L, as $\hat{c}+\widehat{LAC}$. Node A might accept another bearing to L from another pair of neighbors, if it involves less hops than the pair B-C. A then continues the process by forwarding its estimated bearing to L to its neighbors which will help farther away nodes get their estimates for L. Once node A finds its bearings to at least three landmarks that are not on the same line or on the same circle with A, it can infer its position using some triangulation procedure for single-hop scenario.

If the radial method is to be used, a similar argument holds, with the difference that now A needs to know, besides bearings of B and C to L, the radials of B and C from L. If the angle $\widehat{BLN}$ (radial at B; 'N' stands for 'north') is also known, then the angle $\widehat{ALN}$ (radial at A) can also be found since all angles in both triangles are known. The actual downside for this method is in the increased communication - nodes B and C forward two values per landmark (bearing and radial) instead of just one, as in the bearing based method.

To conclude, the method we described infers position and orientation in an ad-hoc network where nodes can measure AOA from communication with their immediate neighbors. The assumption is that all unknown nodes have AOA capability and only a fraction have self positioning capability. Two algorithms were described, DV-Bearing and DV-Radial. The advantages of the method are that it provides absolute coordinates and absolute orientation, that it works well for disconnected networks, and does not require any additional infrastructure. Moreover, since the commu-

nication protocol can be localized, this algorithm is scalable to very large WSNs. Resulted positions have accuracy comparable to the range-based algorithms, and resulted orientations are usable for navigational and tracking purposes. However, high equipment cost makes this method impractical for WSN applications.

2.4 Probabilistic localization techniques

In contrast to deterministic methods [68, 82, 97, 100, 101, 109], *probabilistic* methods [8, 46, 48, 78, 118] take into account uncertainty of the measurements, so given the likelihood of e.g., measured distance, and a prior PDF of the positions of all unknown nodes, they estimate the posterior PDFs of the positions of all unknown nodes. In order to find location estimates, it is sufficient to find either minimum mean square estimate (MMSE), or maximum a posteriori (MAP) of this posterior PDF. These methods are also well-known as *Bayesian* methods. The main drawback of the probabilistic methods is the high complexity, which makes these methods unacceptable in low-power WSNs. Nevertheless, the particle-based approximation [2,32], and appropriate factorization using some message-passing method [77], make probabilistic methods acceptable for localization in WSN.

We provide here general framework for cooperative localization. Let us assume that we have N_s sensors (N_a anchors and N_u unknowns) scattered randomly in a planar region, and denote the 2D location of sensor t by x_t. The unknown node u obtains a noisy measurement d_{tu} of its distance from node t with some probability $P_d(x_t,x_u)$:

$$d_{tu} = \|x_t - x_u\| + v_{tu}\,, \qquad v_{tu} \sim p_v(d_{tu} - \|x_t - x_u\|\,|\Theta_{tu}) \tag{2.21}$$

where, for noise v_{tu}, we can assume a Gaussian distribution p_v (with parameter $\Theta_{tu} = \{\mu_{tu}, \sigma^2_{tu}\}$). However, it is straightforward to change it to any desired distribution, e.g., an empirical distribution obtained by performing the experiments in the deployment area.

The binary variable o_{tu} will indicate whether this observation is available or not:

$$o_{tu} = \begin{cases} 1, & d_{tu}\ observed, \\ 0, & otherwise. \end{cases} \tag{2.22}$$

Finally, each sensor t has some prior distribution denoted $p_t(x_t)$. This prior could be an uninformative one (i.e., with uniform distribution over the whole deployment area) for the unknowns and the Dirac Delta function for the anchors. Then, the

joint posterior PDF is given by:

$$p(x_1, ..., x_{N_u} | \{o_{tu}\}, \{d_{tu}\}) = \prod_{(t,u)} p(o_{tu} | x_t, x_u) \prod_{(t,u)} p(d_{tu} | x_t, x_u) \prod_t p_t(x_t) \quad (2.23)$$

We also need to define probability of detection, from which we can draw variable o_{tu}. For large-scale WSNs, it is reasonable to assume that only a subset of pairwise distances will be available, primarily between sensors which are located within the some radius R. The ideal model of probability of detection is given by:

$$P_d(x_t, x_u) = \begin{cases} 1, & for \ \|x_t - x_u\| \leq R, \\ 0, & otherwise. \end{cases} \quad (2.24)$$

Better approximations of $P_d(x_t, x_u)$ can be obtained using real experiments in the deployment area of interest, and is especially advisable for indoor scenarios. We can also use the exponential model [46], which represents a better approximation:

$$P_d(x_t, x_u) = \exp\left(-\frac{1}{2} \|x_t - x_u\|^2 / R^2\right) \quad (2.25)$$

Our goal is to compute the posterior *marginal* PDF $p(x_t, |\{o_{tu}\}, \{d_{tu}\})$ (for each unknown node t) by marginalizing the joint posterior PDF, which is not tractable for the localization problem. Therefore, we need to factorize the joint posterior PDF using some message-passing method [77]. In addition, due to the presence of nonlinear relationships and potentially non-Gaussian uncertainties, we should use a particle-based approximation [2, 32]. The best known method for this problem is NBP. It is recently used for cooperative localization in the static [46, 48] and the mobile networks [99, 118]. Detailed description of NBP, and a number of extensions will be the main topic of Chapter 3.

2.5 Summary

In this chapter, we reviewed a number of cooperative WSN localization techniques. In particular, we described the standard measurement techniques (RSS, TOA/TDOA, AOA), deterministic localization methods (distance-based and connectivity-based), localization using AOA, and the general framework for probabilistic localization. As we can see, in the state-of-the-art there are a lot of deterministic methods without capability to provide associated uncertainty online, especially in the case of non-Gaussian measurements, Therefore, further investigation of the probabilistic methods will be the main topic of the following chapters of this thesis.

Chapter 3

Message passing methods for cooperative localization in loopy networks

3.1 Introduction

Belief propagation (BP) [77, 121] is a way of organizing the global computation of marginal beliefs in terms of smaller local computations within the graph. It is one of the best-known *message passing* methods for distributed inference in statistical physics, artificial intelligence, computer vision, localization, etc. The whole computation takes a time proportional to the number of links in the graph, which is significantly less than the exponential time that would be required to compute marginal probabilities naively. Therefore, BP is suitable for probabilistic cooperative WSN localization described in Chapter 2. However, due to the presence of nonlinear relationships and non-Gaussian uncertainties, the standard (parametric) BP is undesirable. Nevertheless, a particle-based approximation via *nonparametric* belief propagation (NBP), proposed by Ihler et al. [46, 48], makes BP acceptable for cooperative WSN localization.

However, in loopy networks NBP suffers from similar problems as standard BP, such as inaccurate beliefs and possible non-convergence. Few solutions for this problem will be proposed in this chapter. We start with the description and analysis of the standard BP/NBP techniques, and also modified version of NBP, nonparametric boxed belief propagation (NBBP) [89,93]. Then, we propose four improved message-passing methods for loopy networks: nonparametric generalized belief propagation (NGBP) based on junction tree (NGBP-JT) [90, 96], NGBP based on pseudo-

junction tree (NGBP-PJT) [92], NBP based on spanning trees (NBP-ST) [88, 91], and uniformly-reweighted NBP (URW-NBP) [79, 95, 119, 120].

3.2 Belief propagation (BP)

In the standard BP algorithm [24, 121], the belief at a node t (approximation of the posterior marginal PDF) is proportional to the product of the local evidence at that node $\psi_t(x_t)$, and all the messages coming into node t:

$$M_t(x_t) = k\psi_t(x_t) \prod_{u \in G_t} m_{ut}(x_t) \tag{3.1}$$

where x_t is a state of node t, k is a normalization constant, and G_t denotes the neighbors of node t. The messages are determined by the message update rule:

$$m_{ut}(x_t) = \int_{x_u} \psi_u(x_u)\psi_{tu}(x_t, x_u) \prod_{g \in G_u \setminus t} m_{gu}(x_u)dx_u \tag{3.2}$$

where $\psi_{tu}(x_t, x_u)$ is the pairwise potential between nodes t and u. On the right-hand side, there is a product over all messages going into node u except for the one coming from node t. In other words, the message from node u to node t represents the "opinion" of node u about the location of node t. The messages and beliefs are, of course, represented as PDFs, but not necessarily normalized. In practical computation, one starts with nodes at the edge of the graph, and only computes a message when one has available all the messages required. It is easy to see [121] that each message needs to be computed only once for the graphs without loops.

For cooperative localization, we use an undirected graph [111] $G = (V, E)$ consisting of a set of nodes or vertices V that are joined by a set of edges E. In order to define an undirected graphical model (also knows as Markov random field), we place at each node a random variable x_s taking values in some space. In case of localization, this random variable represents the 2D location, and each edge represents the measured distance. If we exclude the anchor nodes, the graph is obviously undirected.

The relationship between the graph and joint PDF may be quantified in terms of potential functions ψ which are defined over each of the graph's cliques. A clique (C) is a subset of nodes such that for every two nodes in C, there exists an link connecting the two. So the joint PDF is given by:

$$p(x_1, ..., x_{N_u}) \propto \prod_{cliques\, C} \psi_C(\{x_i : i \in C\}) \tag{3.3}$$

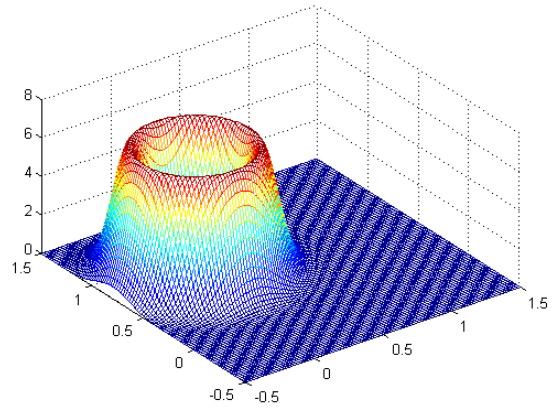


Figure 3.1: Example of pairwise potential $\psi_{tu}(x_t^*, x_u)$ around the anchor node t (placed in the center of the circular distribution).

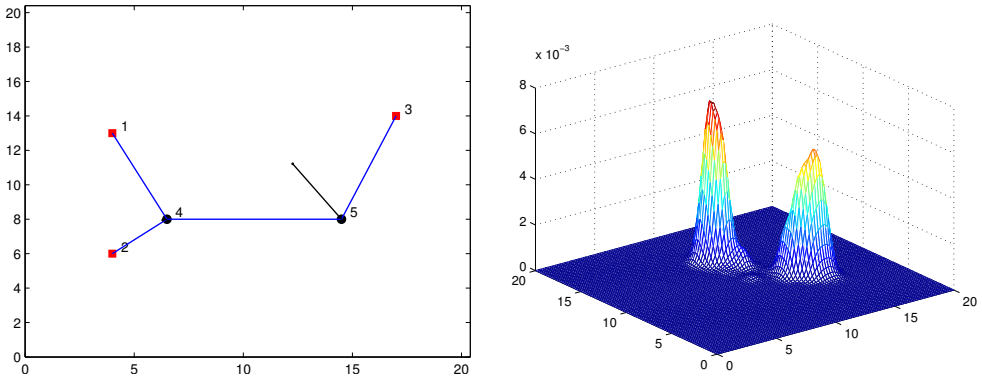
We only need potential functions defined over variables associated with single nodes and pairs of nodes. Single-node potential (prior information about position) at each node t, and the pairwise potential (likelihood function, $p(d_{tu}|x_t, x_u)$) between nodes t and u, are respectively given by:

$$\psi_t(x_t) = p_t(x_t), \tag{3.4}$$

$$\psi_{tu}(x_t, x_u) = \begin{cases} P_d(x_t, x_u)p_v(d_{tu} - \|x_t - x_u\|), & if\ o_{tu} = 1, \\ 1 - P_d(x_t, x_u), & otherwise. \end{cases} \tag{3.5}$$

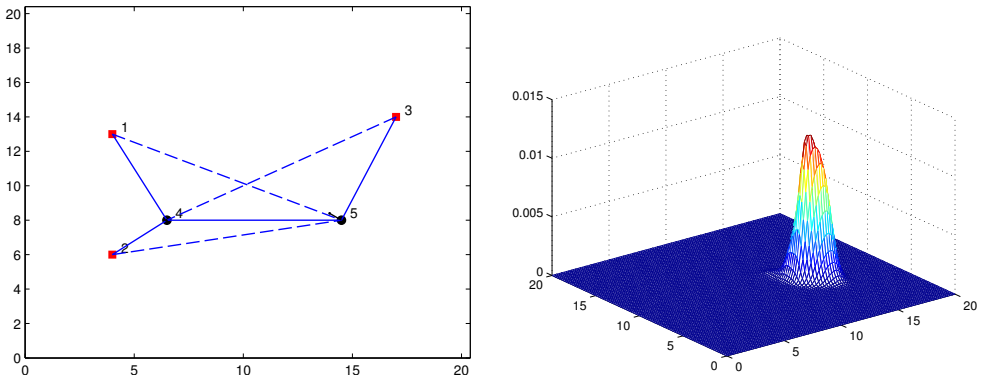
where P_d, and p_v represent probability of detection, and distribution of the measurement noise, respectively (as defined in Section 2.4). Illustration of the pairwise potential between the unknown and anchor node is shown in Figure 3.1. As we can see, it is 2D circular Gaussian distribution around the anchor node t, which provides us information about possible positions of node u.

In addition, it would be useful to exchange information between the nodes which are not directly connected (also known as *negative* information). We define a pair of nodes s and t to be 1-hop neighbors of one another if they observe their pairwise distance d_{st}. Then, we define 2-hop neighbors of node s to be all nodes t such that we do not observe the d_{st}, but do observe d_{su} and d_{ut} for some node u. We can follow the same pattern for the 3-hop neighbors, and so forth. These n-hop neighbors ($n > 1$) contain some information about the distance between them. Therefore, if two nodes do not observe the distance between them, they should be far away from each other. In our case, we will include all 1-hop and 2-hop neighbors, others could be neglected without losing accuracy in the results. This additional information especially helps



(a) (b)

Figure 3.2: (a) An example of 5 node network (nodes 4 and 5 are unknown) and estimated location of node 5 (marked with dot), (b) Belief of node 5. The belief is bimodal because node 5 has only two neighbors.



(a) (b)

Figure 3.3: (a) The network from previous figure which includes additional information from 2-hop neighbors (marked with dashed lines) (b) Belief of node 5. The belief is now unimodal because node 1 "told" to node 5 that it must be far away.

in the bimodal case when, due to the low connectivity (< 3), there are two possible solutions. We illustrate this in 5-node network in Figures 3.2 and 3.3. We used 3 anchors, and 2 unknown nodes. If we use only 1-hop neighbors, the belief of node 4 will be bimodal (i.e., with 2 local maximums) as shown in Figure 3.2b. However, adding 2-hop neighbors will provide an additional information, e.g., node 1 "tells" to node 4 that it must be far away. Now the belief of node 4 has only one mode (Figure 3.3b), so the position estimate is more accurate. Note that we used ideal model for the probability of detection.

Finally, we can write the joint posterior PDF, as function of potentials:

$$p(x_1, ..., x_{N_u} | \{o_{tu}, d_{tu}\}) \propto \prod_t \psi_t(x_t) \prod_{t,u} \psi_{tu}(x_t, x_u) \tag{3.6}$$

Marginalization of the joint posterior PDF (3.6) can be done by applying the BP algorithm. We apply BP to estimate each sensor's posterior PDF, and use the mean value of this marginal (i.e, MMSE estimate) and its associated uncertainty to characterize the sensor positions. We will use a different form of BP algorithm, in order to adapt it to the iterative localization scenario, where it is more practical to compute beliefs in each iteration. This form can be easily found by replacing (3.1) in (3.2). Therefore, each node t computes its belief $M_t^i(x_t)$, the posterior marginal PDF of its 2D position x_t at iteration i, by taking a product of its local potential ψ_t with the messages from its set of neighbors G_t:

$$M_t^i(x_t) \propto \psi_t(x_t) \prod_{u \in G_t} m_{ut}^i(x_t) \tag{3.7}$$

The messages m_{ut}, from node u to node t, are computed by:

$$m_{ut}^i(x_t) \propto \int_{x_u} \psi_{ut}(x_t, x_u) \frac{M_u^{i-1}(x_u)}{m_{tu}^{i-1}(x_u)} dx_u \tag{3.8}$$

In the first iteration of this algorithm, it is necessary to initialize $m_{ut}^1 = 1$ and $M_t^1 = p_t$ for all u, t, and then repeat computation using (3.7) and (3.8) until sufficient convergence. For tree-like graphs, the number of iterations should be at most the length of the longest path in the graph. In case of loopy graphs, there is no such guarantee, but convergence is often achieved after a similar number of iterations.

3.3 Nonparametric belief propagation (NBP)

The presence of nonlinear relationships and potentially non-Gaussian uncertainties in cooperative localization makes standard BP undesirable. However, using particle-based representations via NBP [46,48,105] enables the application of BP to localization in WSN. In NBP, the belief and message update equations, (3.7) and (3.8), are performed using stochastic approximations, in two phases: i) first, drawing particles from the belief $M_t^i(x_t)$, ii) then using these particles to approximate each outgoing message m_{tu}^i.

The main advantage of this approach is the ability to provide information about location estimation uncertainties (in contrast to deterministic approaches), which are not necessarily Gaussian. Furthermore, it is a naturally distributed method, and it converges after a very small number of iterations.

3.3.1 Computing messages

Given N weighted particles $\{W_t^{j,i}, X_t^{j,i}\}$ from the belief $M_t^i(x_t)$ obtained at iteration i, we can compute weighted particles of the outgoing BP message m_{tu}^i. We first consider the case of observed edges (1-hop) between unknown nodes. The distance measurement d_{tu} provides information about how far sensor u is from sensor t, but no information about its relative direction (see Figure 3.1). To draw a particle of the message ($x_{tu}^{j,i+1}$), given the particle $X_t^{j,i}$ which represents the position of sensor t, we simply select a direction $\theta^{j,i}$ at random ($i = 1$), uniformly in the interval $[0, 2\pi)$. However, starting from the second iteration, we can also include angular information [48] from the previous iteration using already computed beliefs. We then shift $X_t^{j,i}$ in the direction of $\theta^{j,i}$ by an amount which represents the estimated distance between nodes u and t ($d_{tu} + v^j$):

$$x_{tu}^{j,i+1} = X_t^{j,i} + (d_{tu} + v^j)[\sin(\theta^{j,i}) \;\; \cos(\theta^{j,i})] \tag{3.9}$$

For example, if node t is an anchor, the unknown node u, is located on noisy circle around node t. That means that the particles from node u are distributed according to distribution shown in Figure 3.1. Assuming that there is detection between sensor nodes t and u with probability $P_d(x_t, x_u)$, the particles are weighted by the reminder of the message-update rule (3.8):

$$w_{tu}^{j,i+1} = P_d(X_t^{i,j}, x_u) \frac{W_t^{j,i}}{m_{ut}^i(X_t^{i,j})} \tag{3.10}$$

As we can see, for the denominator of (3.10), we need to know the parametric form of the message. We can approximate it using *kernel density estimate* (KDE)[1]. The optimal value for bandwidth h_{tu}^{i+1}, can be obtained in a number of ways. The simplest technique is to apply the "rule of thumb" estimate [104]:

$$h_{tu}^{i+1} = N^{-\frac{1}{3}} \, Var(\{x_{tu}^{i+1}\}) \tag{3.11}$$

It is also necessary to define messages coming from anchor nodes, which can be found using (3.8) and the belief of the anchor node x_t^* ($M_t^i(x_t) = \delta(x_t - x_t^*)$):

$$m_{tu}^{i+1}(x_u) \propto \psi_{tu}(x_t^*, x_u) \tag{3.12}$$

[1] Approximation of the distribution $p(x)$: $\hat{p}(x) = \sum_j w^j K^h(x - x^j)$ given a Kernel $K^h(x)$. The most common kernel function (K^h) is the spherically symmetric Gaussian kernel: $K^h(x) = N(x, 0, hI)$, where bandwidth h controls the variance [48, 104].

The messages along unobserved edges (2-hop, 3-hop ...) should be represented in a parametric form since their potentials have the form $1 - P_d(x_t, x_u)$ which is typically not normalizable (because it tends to 1 as the distance becomes large). Using the message-update rule (3.8), pairwise potential (3.5) for $o_{tu} = 0$, and particles from the belief M_t^i, an estimate of the outgoing message to node u is given by:

$$m_{tu}^{i+1}(x_u) = 1 - \sum\nolimits_j \frac{W_t^{j,i}}{m_{ut}^i(X_t^{i,j})} P_d(X_t^{j,i}, x_u) \tag{3.13}$$

Finally, the messages along unobserved edges coming from anchor nodes ($W_t^{j,i} = 1/N$) are given by:

$$m_{tu}^{i+1}(x_u) = 1 - P_d(x_t^*, x_u) \tag{3.14}$$

3.3.2 Computing beliefs

To estimate the belief $M_u^{i+1}(x_u)$ using (3.7), we draw particles from the product of several KDEs of the messages and eventual analytic messages ((3.13) and (3.14)). Since it is very difficult to draw particles from the product, we use the *proposal* distribution $q_u^{i+1}(x_u)$, the sum of the messages, and then reweight all particles. This procedure is well-known as *mixture importance sampling* (MIS) [48].

Denote the set of neighbors of u, having observed edges to u excluding anchors, by G_u^0, and the set of of all neighbors by G_u. In order to draw N particles, we create a collection of kN weighted particles (where $k \geq 1$ is a parameter of the sampling algorithm) by drawing $kN/\left|G_u^0\right|$ particles from each message m_{tu} (with $t \in G_u^0$) and assigning each particle a weight equal to the ratio:

$$W_u^{j,i+1} = \frac{\prod_{v \in G_u} m_{vu}^{i+1}(X_u^{j,i+1})}{q_u^{i+1}(X_u^{j,i+1})} \tag{3.15}$$

where the proposal distribution $q_u^{i+1}(X_u^{j,i+1})$ is given by:

$$q_u^{i+1}(X_u^{j,i+1}) = \sum\nolimits_{v \in G_u^0} m_{vu}^{i+1}(X_u^{j,i+1}) \tag{3.16}$$

Some of these calculated weights are much larger then the rest, especially after more iterations. This means that any particle-based estimate will be dominated by the influence of a few of the particles, and the estimate could be erroneous. To avoid this, we then draw N values independently from the collection $\{W_t^{j,i+1}, X_t^{j,i+1}\}$ with probability proportional to their weight, using *resampling with replacement* [2,10,64]. This means that we create N equal-weight particles drawn from the product of all incoming messages.

3.3.3 Convergence of NBP

A node is located when a convergence criteria is met. We can use Kullback-Leibler (KL) divergence [48], a common measure of difference between two distributions. For the particle based beliefs in NBP algorithm, KL-divergence between beliefs in two consecutive iterations, is given by:

$$KL_u^{i+1} = \sum_j W_u^{j,i+1} \log \frac{W_u^{j,i+1}}{M_u^i(X_u^{j,i+1})} \tag{3.17}$$

Note that the set of particles is different in two consecutive iterations, so we had to use the parametric form of the belief from the previous iteration (denominator in (3.17)) computed at the particles from the current iteration. When KL_u^{i+1} drops below the predefined threshold, the node u is located and starts to behave as an anchor. In this way, we can locate all nodes incrementally. The execution is over when KL drops below the threshold for all nodes, or when the maximum number of iteration is reached. In any case, the estimated positions of all unknowns and their uncertainties will be available. However, note that the number of iterations can be easily predefined, given the structure of the graph (e.g., the communication radius, and the diameter of the deployment area).

3.3.4 Nonparametric boxed belief propagation (NBBP)

We propose NBBP [89], a novel variant of NBP algorithm. The main goal is to increase the performance of the algorithm by adding boxes which constraint the area from which the particles are drawn. These boxes, also called *bounded boxes* [8], which are created almost without any additional communication between nodes, are also used to filter erroneous particles in each iteration. In order to decrease the computational and the communication cost, we also use an incremental approach by locating the node as soon as convergence criterion is satisfied.

The following modifications are added to the already described NBP algorithm:

- Initial particles are drawn from bounding box that covers the region where the anchors' ranges overlap (Figure 3.4).
- In each iteration, erroneous particles of the messages and beliefs (all the particles which are outside of the appropriate box) are filtered out.
- Nodes are located in an incremental way: As soon as the belief sufficiently converges (according to (3.17)), we characterize the sensor positions with mean value and uncertainty, and from that point we consider this node as an anchor.

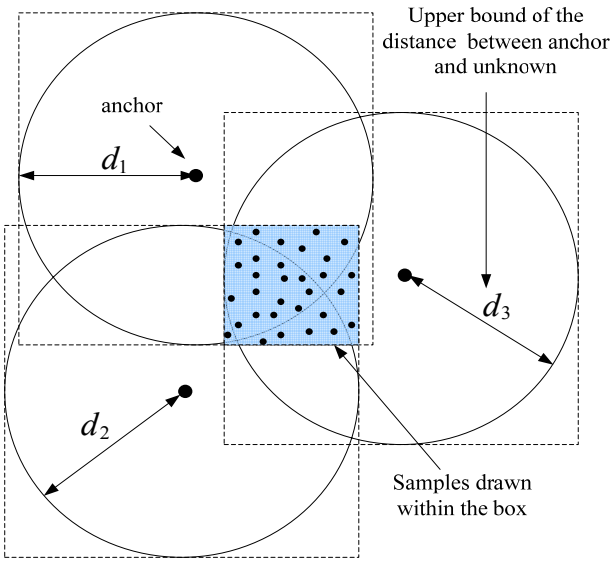
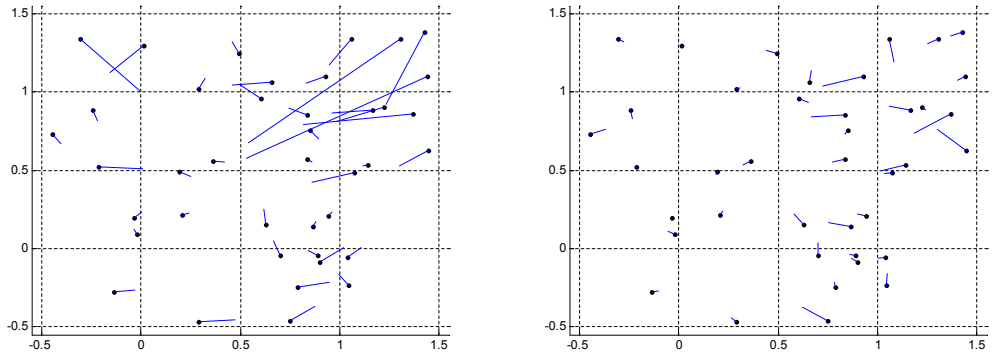


Figure 3.4: Drawing particles within the box that covers the region where anchors' ranges overlap.

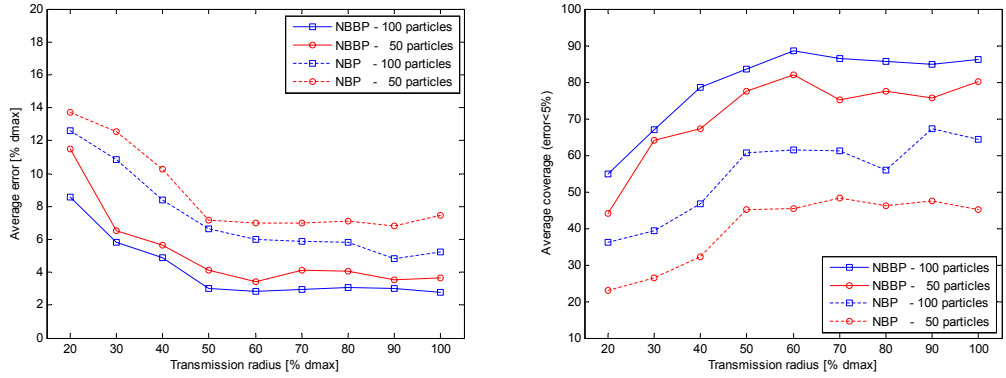


(a) (b)

Figure 3.5: Comparison of the results for a 50-node network (a) NBP, (b) NBBP. The line between the true and the estimated positions represents the error.

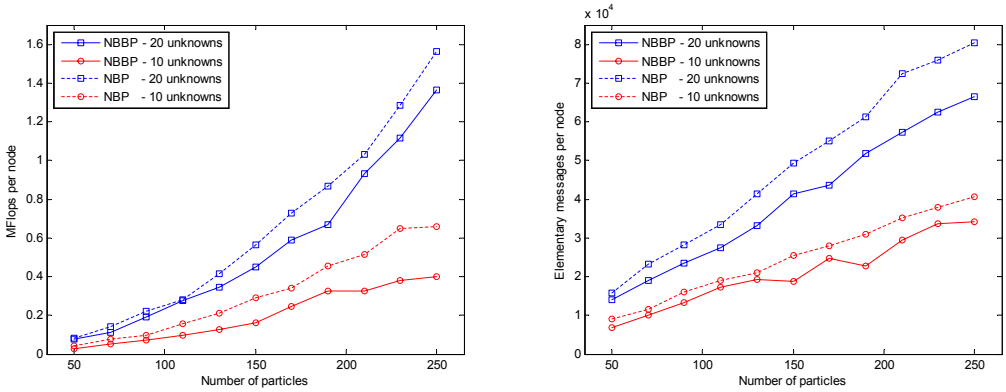
3.3.5 Simulation results

We assume that there are 50 nodes randomly deployed in 2 x 2 area, 40 of them are the unknowns. The values of parameters are set as follows: standard deviation of the Gaussian noise of the measured distance ($\sigma = 0.1$), transmission radius ($R =$ 30% of the diameter of the deployment area: $d_{\max} = 2\sqrt{2}\,\mathrm{m}$), number of particles ($N = 50$ and $N = 100$), and KL threshold ($KL_{\min} = 0.02$). The error is defined as Euclidian distance between the true and estimated location. Finally, each point in the simulations represents the average over 20 Monte Carlo runs.



(a) (b)

Figure 3.6: Comparison of the (a) accuracy and (b) coverage



(a) (b)

Figure 3.7: Comparison of the (a) computational and (b) communication cost

Using the defined scenario, we compared both algorithms (NBP and NBBP). Obtained the results shown in Figure 3.5. As expected, the location estimates for the NBBP are more accurate since all the estimates are placed within their bounded boxes, which limit the maximum error. Moreover, since the nodes near the edges suffer from low connectivity, the error for those nodes is higher.

In Figure 3.6a and 3.6b, we compare the average error and coverage with respect to the transmission radius. The coverage is defined as a percentage of located nodes with error less than predefined tolerance. In our case we set it to 5%, but this is an application dependent parameter. As we can see, the accuracy is increasing as transmission radius is increasing. For high values of the transmission radius, the accuracy and the coverage are nearly constant since the nodes start to receive redundant information caused by high connectivity. Moreover, NBBP consistently

outperforms NBP for each considered value of R, and the number of particles also significantly affects the performance of both methods.

Finally, in Figure 3.7, we show a comparison of the computational and communication cost with respect to the number of particles for two different densities (we use 20 and 10 unknown nodes, respectively). To measure the communication cost, we count *elementary messages*, where one elementary message is defined as one scalar value (e.g., one coordinate of the particle). For this analysis, we assume that there is no compression of the data. The main conclusion is that the NBBP algorithm performs better than NBP in all terms. This result is expected because constructed boxes and filtering in each iteration increase accuracy, and the incremental approach decreases the computational and communication cost.

3.3.6 Comparison with MDS

In order to compare NBBP with deterministic methods, we choose the best representative. It has been already shown [61,101] that MDS outperforms other well-known deterministic methods (DV-distance, multilateration, etc.). Therefore, we will compare the NBBP with variants of MDS (MDS-MAP, MDS-MAP-P, and MDS-MAP-PR) described in Chapter 2. To that end, we reuse scenario from [101]. We consider two networks in 10m x 10m area : i) random uniform network with 200 nodes (190 unknowns and 10 anchors), and ii) irregular C-shape network with 160 nodes (150 unknowns and 10 anchors). The measured distance has zero-mean Gaussian distribution with standard deviation set to 5% of the true distance ($\sigma_{tu} = 0.05 d_{tu}$). Transmission radius varies from 1.25m to 2.5m. As error metric, we use median error (50th percentile). For NBBP, we use $N = 100$ particles and $N_{iter} = 10$. We averaged results over 10 Monte Carlo runs.

The results are shown in Figure 3.8. As we can see, MDS-MAP performs the worst in both cases, because the distance between non-neighboring nodes is approximated with the shortest path distance [101]. This is very coarse approximation, especially in irregular networks. Moreover, we can see that MDS-MAP-PR performs slightly better than MDS-MAP-P thanks to the refinement (least-square minimization). NBBP performs worse than these two methods in the random uniform network, but better in the C-shaped network. Obviously, since the noise is Gaussian, MDS methods provide close-optimal solution if the network is regular. However, if the network is irregular, approximation of the n-hop (n>1) distances is very coarse even for small local maps in MDS-MAP-P. On other hand, NBBP is robust to network topologies thanks to the fully distributed nature, in which only messages from

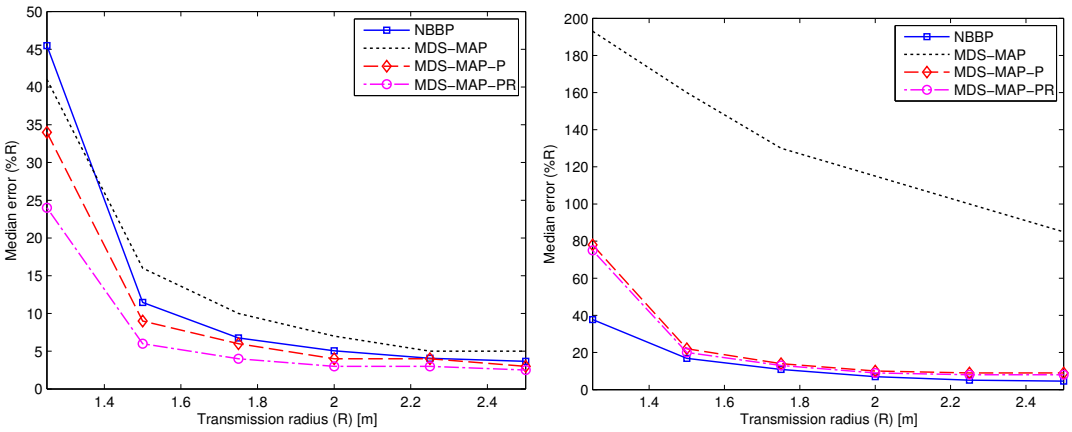


Figure 3.8: Comparison between NBBP and MDS for (a) random uniform, and (b) C-shape networks.

the local neighbors have been used. This is the reason why NBBP[2] performs the best for C-shaped network, especially for small values of transmission radius.

3.4 Correctness of BP

The BP algorithm, defined by (3.1) and (3.2) in previous section, does not make a reference to the topology of the graph that it is running on. Thus, there is nothing to stop us from implementing it on a graph that has loops. One starts with some initial set of messages, and simply iterates the message-update rules (3.2) until they eventually converge, and then one can read off the approximate beliefs from the belief equations (3.1). But if we ignore the existence of loops and permit the nodes to continue communicating with each other, messages may circulate indefinitely around these loops, and the process may not converge to a stable equilibrium. One can find an examples of graphical models with loops, where, for certain parameter values, the BP algorithm fails to converge or predicts beliefs that are inaccurate. On the other hand, the BP algorithm can be successful in graphs with loops, e.g. error-correcting codes defined on Tanner graphs that have loops [37].

Let us consider the example network in Figure 3.9. In this network, there are 3 unknown nodes (A, B and C) and 3 anchor nodes (E_A, E_B, and E_C) which represent the local evidence. The message-passing algorithm (BP) can be thought of as a way of communicating local evidences between nodes such that all nodes calculate their beliefs given all the evidence.

[2]To simplify notation in the following part of the thesis, we will assume that the bounded boxes are part of the standard NBP and all proposed NBP-based methods.

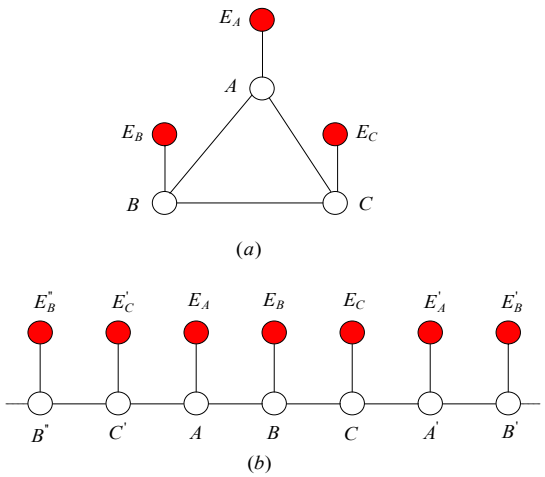


Figure 3.9: (a) A simple loopy network, (b) Corresponding unwrapped network for the first 3 iterations

In order for BP to be successful, it needs to avoid *double counting* [77, 114] - a situation in which the same evidence is passed around the network multiple times and mistaken for new evidence. Of course, this is not possible in tree-like networks because when a node receives some evidence, it will never receive that evidence again (see (3.1) and (3.2)). In a loopy network double counting can not be avoided. For example in Figure 3.9a, node B will send A's evidence to C, but in the next iteration, C will send that same information back to A. Thus, it seems that BP in such a network will give the wrong answer.

However, BP could still lead to correct inference if all evidence is "double counted" in equal amounts. This could be formalized by an unwrapped network corresponding to the loopy network. The unwrapped network is a tree-like network constructed such that performing BP in the unwrapped network is equivalent to performing BP in the loopy network. The basic idea is to replicate the nodes as shown in Figure 3.9b. For example, the message received by node B after 3 iterations of BP in the loopy network are identical to the final messages received by node B" in the unwrapped network. In this way, we can create infinite network. The importance of the unwrapped network is that since it is tree-like, BP on it is guaranteed to give the correct beliefs. However, the usefulness of these beliefs depends on the similarity between the PDF induced by the unwrapped problem and the original loopy problem. And this similarity is satisfied in single-loop network after a finite number of iterations. In the general case, BP will converge when the addition of these additional nodes at the boundary will not alter the posterior PDF of the node in the center.

Finally, in Gaussian networks [115] the factor that determines the goodness of the

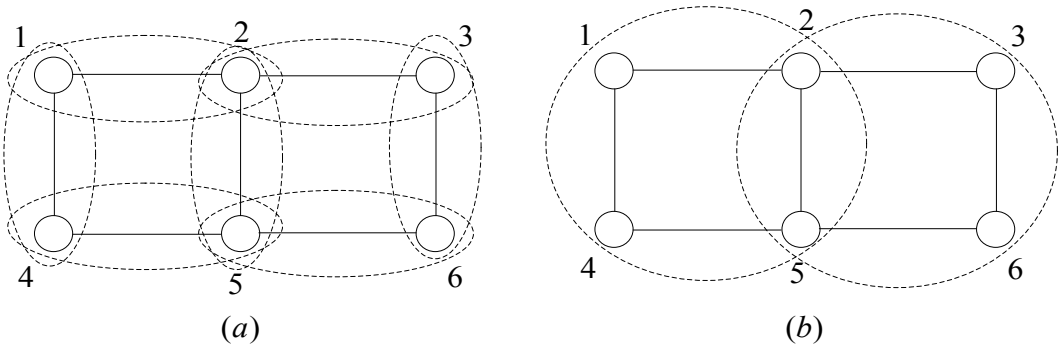


Figure 3.10: The basic clusters in the (a) Bethe approximation and (b) Kikuchi approximation

approximation and the convergence rate is the amount of statistical independence between the root nodes and the leaf nodes in the unwrapped network. In Gaussian networks with multiple loops the mean at each node is guaranteed to be correct but the confidence around that mean may be incorrect (usually, overconfident). These results give a theoretical justification for applying BP in certain networks with multiple loops. This may enable fast, approximate probabilistic inference in a range of new applications, where measurement error could be similar to the Gaussian model. For an extensive analysis of this topic, we refer the reader to [48, 66] where many useful theorems are provided.

In following sections, we provide several solutions for cooperative localization in loopy graphs.

3.5 Generalized belief propagation (GBP) methods

In standard BP, all messages are always going from a single node to another single node. It is natural to expect that messages from groups of nodes to other groups of nodes could be more informative, and thus lead to better inference. That is the basic idea behind GBP methods, which are the main topic of this section.

3.5.1 GBP based on Kikuchi approximation (GBP-K)

We start with brief description of GBP-K [121], in which the nodes are clustered into regions (also know as *Kikuchi* approximation), and then performed GBP. Standard BP is a special case in which each pair of neighboring nodes represent one region (also known as *Bethe* approximation). In Figure 3.10, we show the basic clusters for both approximations.

GBP-K algorithm nearly always improves, at least slightly, over the performance

of standard BP, and it can significantly outperform standard BP if the graphical model under consideration has only short loops. However, the complexity of GBP-K grows exponentially with the size of the basic clusters that are chosen. If we include all loops as the basic clusters, the GBP-K algorithm is exact, but computationally unacceptable. This is the reason why this method is not appropriate for the localization problem at hand. For the detailed description of the GBP-K method and its variations, the reader is referred to [121].

3.5.2 GBP based on junction-tree (GBP-JT)

GBP-JT is a standard method for exact inference in graphical model. That means that all posterior marginals will provide the true information about uncertainty of node estimates. This method can be derived using the *elimination* procedure [52]. The graph is first *triangulated* (i.e. "virtual" edges are added so that every loop of length more than 3 has a chord). Given a triangulated graph, with cliques C_i and potentials $\psi_{C_i}(x_{C_i})$, and given the corresponding junction tree (JT), which defines links between the cliques, we send the following message from clique C_i to clique C_j by the message update rule:

$$m_{ij}(x_{S_{ij}}) = \sum_{C_i \setminus S_{ij}} \psi_{C_i}(x_{C_i}) \prod_{k \in G_i \setminus j} m_{ki}(x_{S_{ki}}) \tag{3.18}$$

where $S_{ij} = C_i \cap C_j$, and where G_i are the neighbors of clique C_i in the JT. The belief at clique C_i is proportional to the product of the local evidence at that clique and all the messages coming into clique i:

$$M_i(x_{C_i}) = k\psi_{C_i}(x_{C_i}) \prod_{j \in G_i} m_{ji}(x_{S_{ji}}) \tag{3.19}$$

Beliefs for single nodes can be obtained via further marginalization:

$$M_i(x_i) = \sum_{C_i \setminus i} M_i(x_{C_i}) \quad \text{for } i \in C_i \tag{3.20}$$

Equations (3.18), (3.19), and (3.20) represent GBP-JT algorithm which is valid for arbitrary graphs. Note that in case of continuous variables, summation have to be replaced by integration. The BP algorithm defined with (3.1) and (3.2) is a special case of GBP-JT, obtaining by noting that the original tree is already triangulated, and has only pairs of nodes as cliques. In that case, sets S_{ij} are single nodes, and marginalization using (3.20) is unnecessary.

We also need to define *clique potential* ψ_{C_i}, which is given as a product of all

single-node and pairwise potentials. The potentials of 2-node clique $C_i(t,u)$ and 3-node clique $C_j(t,u,v)$ are, respectively, given by:

$$\psi_{C_i}(x_{C_i}) = \psi_{tu}(x_t, x_u)\psi_t(x_t)\psi_u(x_u) \tag{3.21}$$

$$\psi_{C_j}(x_{C_j}) = \psi_{tu}(x_t, x_u)\psi_{tv}(x_t, x_v)\psi_{uv}(x_u, x_v)\psi_t(x_t)\psi_u(x_u)\psi_v(x_v) \tag{3.22}$$

The potentials of larger cliques can be defined in analog way. The single-node potential ψ_t and pairwise potential ψ_{tu} are given by equations (3.4) and (3.5), respectively. If the edge (t,u) is triangulated, we set $\psi_{tu} = 1$. To provide better understanding of GBP-JT, we analyze an example in Appendix A.

As in case of BP, because of the presence of nonlinear relationships, and potentially highly non-Gaussian uncertainties, we need to apply nonparametric approximation of GBP-JT method (NGBP-JT). We skip the description since the general framework will be provided for the NGBP-PJT method in Section 3.5.4 (which uses the same nonparametric approximation as NGBP-JT). Detailed example can be also found in [90].

3.5.3 GBP based on pseudo-junction-tree (GBP-PJT)

There remained two main problems of GBP-JT method: i) how to efficiently form the JT in an arbitrary network, and ii) how to decrease the effective dimensionality of the particles. To address these problem, we propose GBP-PJT. The main difference comparing with GBP-JT is the formation of pseudo-junction tree (PJT), which represents the approximated JT based on thin graph (TG). In addition, in order to decrease the number of high-dimensional particles, we use improved importance density function, and also propose dimensionality reduction of the messages. As by-product, we also propose NBP based on TG (NBP-TG), cheaper variant of NBP, which runs on the same graph as NGBP-PJT.

JT formation

JT is a clique tree based on *triangulated* graph [52, 55], i.e., a graph with additional "virtual" edges so that every loop of length more than 3 has a chord. In triangulated graph, each 3-node loop (which is not subset of any larger clique) represents 3-node clique, and each edge (which is not subset of any 3-node clique) represents 2-node clique. Larger cliques (> 3) should be avoided, but this is not possible in most graphs, even with optimal triangulation procedure. Using these cliques as hypernodes, we can define a *cluster graph* [55] by connecting each pair of the cliques with minimum one common node (i.e., non-empty intersection). Using cluster graph,

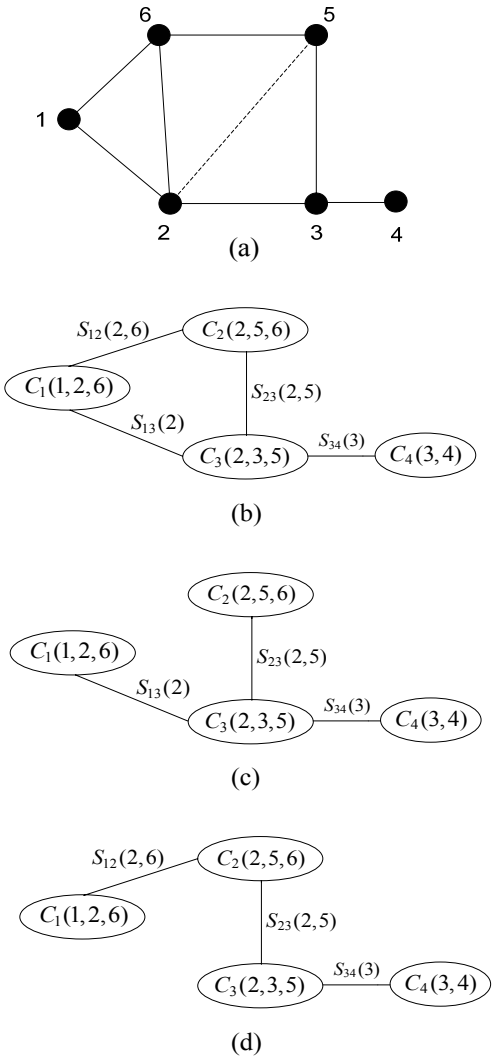


Figure 3.11: (a) Triangulated 6-node graph, and corresponding (b) cluster graph, (c) clique tree, and (d) JT.

we can create a lot of clique trees, but just very few of them represent the JT. The JT is a *maximum spanning tree* of the cluster graph, with weights given by the cardinality of the intersections between cliques. It is already proved [55] that this is a way to satisfy the main property of the JT, the *running intersection property* (RIP). The RIP is satisfied if and only if each node, which is in two cliques C_i and C_j, is also in all cliques on the unique path between C_i and C_j. If the RIP is not satisfied for any node, there is no theoretical guarantee that its belief in one clique is the same as its belief in another clique.

We illustrate the whole procedure in Figure 3.11. We first triangulate the graph

by adding the edge between nodes 2 and 5 (Figure 3.11a). Then we form the cluster graph (Figure 3.11b) with cliques $C_i(t, u, v)$ and separator sets $S_{ij}(q, r)$ ($S_{ij} = C_i \cap C_j$), where t, u, v are the nodes in the clique, and q, r are the *separator nodes*. Finally, any spanning tree (ST) represents the clique tree like in Figure 3.11c and Figure 3.11d. The tree in Figure 3.11d is maximum ST ($|S_{12}| > |S_{13}|$), so it represents the JT of the initial graph. Note that the tree in Figure 3.11c does not satisfy RIP since the node 6, which is in C_1 and C_2, is not in C_3.

The described procedure represents the exact formation of the JT, also known as *chordal graph* method. The main problem of this approach is the triangulation phase. Finding, a minimum triangulation, i.e., one where the largest clique has minimum size, is *NP*-hard problem due to the number of permutations that must be checked. Of course, there exist approximate methods (e.g., [45]) which are less expensive, but, according to authors, still too costly. For more details, see Chapter 10 in [55].

PJT formation

Due to the high complexity of the optimal JT formation, it is necessary to find some approximation that will be suitable for the localization problem. Therefore, we try to achieve the following:

(a) The number of cliques should be reasonable (i.e., in the order of the number of nodes).

(b) In order to reduce the dimensionality of the problem, each clique will include no more than 3 nodes.

(c) Since the triangulation is expensive procedure, we are going to avoid it, even if it causes the break of RIP for the small percentage of the nodes.

After these approximations, the final result represents, strictly speaking, the clique tree. However, since it is very close to the JT (measured by percentage of the nodes that satisfies RIP), we name it *pseudo-junction tree* (PJT).

In order to satisfy the conditions (a) and (b), we need to decrease the number of edges in the graph by formation of *thin graph* (TG). That can be easily done using modified version of *breadth first search* (BFS) method. Standard BFS method [7] begins at randomly chosen root node and explores all the neighboring nodes. Then each of those neighbors explores their unexplored neighbors, and so on, until all the nodes are explored. In this way, there will not be a loop in the graph because all the nodes will be explored just once. Thus, the final result of BFS is a ST. The worst

Algorithm 3 Searching for TG and cliques using modified BFS method

Input: node list Q and root node $root$
Copy to node lists: $Nodes, NewVisit \leftarrow Q$
Set current root: $r \leftarrow root$
Create list of neighbors for all nodes $n \in Q$: G_n
while $Nodes$ is not empty **do**
    for all nodes $t \in G_r$ **do**
        if $t \in Nodes$ **then**
            Remove t from $Nodes$
            Insert t in $WaitingRoots$
            Insert d_{rt} in T
        else if $d_{rt} \notin T$ *and* $r \in NewVisit$ **then**
            Insert d_{rt} in T
            Remove r from $NewVisit$
            Create 3-node cliques:
            for all $q \in PreviousRoots$ **do**
                if $\{d_{rq}, d_{tq}\} \in T$ **then**
                $C^{3nodes} \leftarrow \{r, t, q\}$
                end if
            end for
        end if
    end for
    Insert r in $PreviousRoots$
    Set current root: $r \leftarrow$ first unused node from $WaitingRoots$
end while
Create 2-node cliques C^{2nodes}: each edge in T which is not subset of C^{3nodes}
Output: thin graph $\{Q, T\}$ and cliques $C = C^{2nodes} \cup C^{3nodes}$

case complexity is $O(v + e)$, where v is the number of nodes and e is the number of edges in the graph, since every node and every edge will be explored in the worst case.

Nevertheless, a ST is very coarse approximation of the original graph since it excludes a lot of edges from the graph. For example, in any ST, one communication failure breaks the graph into the two parts. As a consequence, we need more STs in order to have reasonable accurate inference in graphical models (this idea will be used in Section 3.6). Therefore, we modify standard BFS method by permitting each root node to make additional visit to the node that was already visited by some of the previous roots. All edges found by the first and the second visit, together with all the nodes from the original graph, represent the TG. In addition, the second visit will automatically form a loop, so we use it to form 3-node clique. The 2-node cliques can be found easily by taking all the edges that appear in TG, which are not already subset of any a 3-node clique. The worst complexity is $O(v+e+v\cdot(v-1)) \approx O(v^2)$,

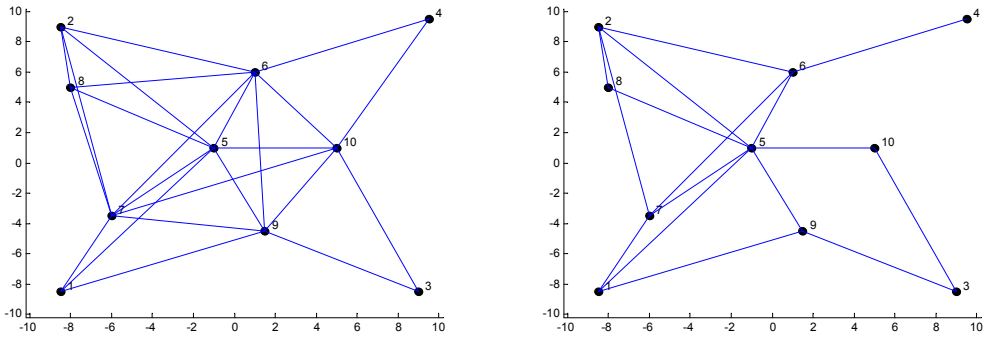
Algorithm 4 PJT formation using Prim's algorithm

1: **Input**: node list Q and cliques C
2: Create weighted cluster graph:
3: **for all** pairs $\{i, j\}$ **do**
4: $\quad Weights(i, j) = |C_i \cap C_j|$
5: **end for**
6: Insert random root clique in $CurrentList$
7: **while** $|CurrentList| < |C|$ **do**
8: $\quad$ Choose edge $\{m, n\}$ with maximal weight, such that C_m is in $CurrentList$ and C_n is not
9: $\quad$ Insert C_n in $CurrentList$
10: $\quad$ Insert edge $\{m, n\}$ in D
11: **end while**
12: **Output**: Pseudo-junction tree $\{C, D\}$

since for each of the additional visits we need to check all previous roots (all the nodes but one, in worst case). The detailed pseudocode is shown in Alg. 3, and an example of original graph and corresponding TG are shown in Figure 3.12a and Figure 3.12b, respectively.

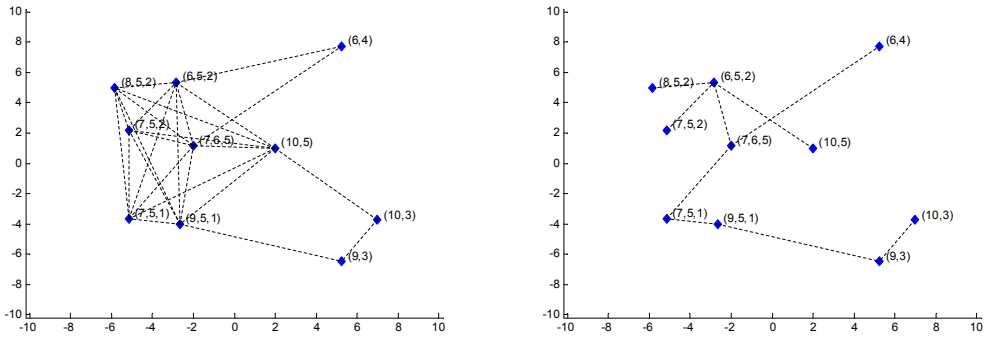
The main benefit of the TG is that it mainly includes 3-node loops. The number of these loops, which is obviously always less than number of nodes, is nearly constant with respect to connectivity, so the number of cliques will be nearly constant as well. On the other hand, the main drawback is that there exist the loops which include more that 3 nodes, but just very few of them. These loops should be triangulated, but we prefer to avoid it in order to keep reasonable complexity. Thus, for n-node loops ($n > 3$), we form maximum n 2-node cliques, using each edge (which is not already subset of any 3-node clique) of the loop as a clique. Another potential problem are the nodes with less than three neighbors. However, these nodes can be still located since we bounded the estimate within its bounding box (see Section 3.3.4), created using original (not thin) graph. Moreover, if possible, all leaf nodes (with low connectivity) should be as close as possible to the anchor nodes.

Having defined cliques, we can form the cluster graph by connecting all pairs of the cliques with non-empty intersection (see Figure 3.13a). As we already mentioned, the JT, as well as PJT, is the maximum ST of the cluster graph. It can be found using e.g., Prim's algorithm [113], as shown in Alg. 4. The Prim's algorithm is a method that finds a maximum (or minimum) ST for a connected weighted undirected graph. That means that the total weight of all the edges in the final tree is maximized (or minimized). In our case, the algorithm starts with a list (i.e., *CurrentList* in Alg. 4) which initially includes only randomly chosen root clique. At each step, among all the edges between the cliques in the list and those not in the list yet, it chooses



(a) (b)

Figure 3.12: (a) Example of 10-node graph, and (b) corresponding TG. The initial root is node 1.



(a) (b)

Figure 3.13: (a) Cluster graph based on TG from Figure 3.12b, and (b) corresponding PJT (maximum ST of given cluster graph). Each clique is placed at the centroid of its node positions. The root clique is (10,5).

the one with maximum weight and increases the list by adding the explored clique. Finally, it stops when all the cliques are spanned. The example of PJT is shown in Figure 3.13b. The worst case complexity is $O(e \cdot log(v))$ [113], but in our case the weights are binary ($|S_{ij}| = 1$, or $|S_{ij}| = 2$), so it will be significantly faster.

The BP/GBP methods are naturally distributed through the graph which means that there is no central unit which will handle all computations. Thus, the proposed PJT formation should be done in a distributed way. It is already well-known that there are a number of distributed techniques to form any ST, which can be reused for formation of the TG. For more details, we refer the reader to [65, 123].

Having defined all cliques, it remains to define the communication between neighboring cliques. Since the separator sets, between each pair of neighboring cliques,

are always non-empty, the separator nodes are responsible to perform the communication. Practically, these nodes represents the cluster heads. For example, in Figure 3.13b, the node 3 will request all the data from node 9, and upon receiving, it will send the data to node 10, and vice versa.

Finally, it is important to note that the approximations we made usually break the RIP for some small number of nodes. For instance, in the PJT in Figure 3.13b, the node 10 (due to the non-triangulated 4-node loop: 3-9-5-10), and node 7 (due to the appearance of 4-node clique: 2-6-5-7) do not satisfy the RIP. Therefore, we do not have a guarantee that the belief of that node in one clique is the same as its belief in another clique [55]. Nevertheless, for the localization, this is not a problem since we use the bounded boxes (see Section 3.3.4) for the initial set of particles.

3.5.4 Nonparametric approximation of GBP-PJT method

In this section, we propose an efficient nonparametric approximation of GBP-PJT method (which is also valid for GBP-JT). We first adapt GBP-JT to iterative scenario for cooperative localization, so the equations (3.18), (3.19), at iteration $m+1$, can be written as:

$$m_{ij}^{m+1}(x_{S_{ij}}) = \frac{1}{m_{ji}^{m}(x_{S_{ji}})} \sum_{C_i \setminus S_{ij}} M_i^m(x_{C_i}) \tag{3.23}$$

$$M_i^{m+1}(x_{C_i}) \propto \psi_{C_i}(x_{C_i}) \prod_{j \in G_{C_i}} m_{ji}^{m+1}(x_{S_{ji}}) \tag{3.24}$$

At the beginning, it is necessary to initialize $m_{ij}^1 = 1$, and $M_i^1 = \psi_{C_i}$. Instead of running on JT, we are going to run this method on PJT. We apply nonparametric approximation through 3 main phases: i) drawing initial particles from the clique potentials, ii) computing messages, and iii) computing beliefs.

Drawing particles from the cliques

Let us draw N_C weighted particles, $\{W_{C_i}^{k,m}, X_{C_i}^{k,m}\}$ $(k = 1, ..., N_C; m = 1)$, from clique i. Since it is computationally very expensive to draw particles from $M_i^1 = \psi_{C_i}$, we need to find appropriate importance density function. Thus, for the initial particles, we are going to use two constraints: i) each particle of the node must be inside its bounding box, and ii) the distance between each pair of the nodes in clique should be close to the mean value of the measured distance. Taking this into account, our importance density function $q_{C_i}^m$ (for $m = 1$) for clique $C_i(t, u)$ is given

by[3]:

$$q^1_{C_i}(x_{C_i}) = q^1_{tu}(x_t, x_u) = \begin{cases} \psi_t(x_t)\psi_u(x_u),\ if\ \|\mu_{tu} - \|x_t - x_u\|)\| < \delta \\ \epsilon,\ otherwise \end{cases} \tag{3.25}$$

where μ_{tu} is the mean value of measured distance. The parameter δ should be chosen so as to encompass nearly the whole PDF. Otherwise, if we cut out significant part of the PDF, the final beliefs will be overconfident. For instance, if p_v is a Gaussian with standard deviation σ_d, $\delta = 3\sigma_d$ could be a good choice since it will encompass about 99% of the PDF. If the constraint is not satisfied, there is very small probability ($\epsilon \to 0$) for the particle in that area, so we can neglect it. Finally, it is straightforward to show (using (3.25)) that the importance density function, for 3-node clique $C_j(t, u, v)$, can be found as:

$$q^1_{C_j}(x_{C_j}) = \sqrt{q^1_{tu}(x_t, x_u) q^1_{tv}(x_t, x_v) q^1_{uv}(x_u, x_v)} \tag{3.26}$$

To draw clique particle, we need to draw node particles within its boxes and accept the particle if the constraint is satisfied. If not, we reject the particle, and try again. The weights of the particles can be easily computed by:

$$W^{k,1}_{C_i} = \frac{\psi_{C_i}(X^{k,1}_{C_i})}{q^1_{C_i}(X^{k,1}_{C_i})} \tag{3.27}$$

Then, these weights (as well as all computed weights in the following text) have to be normalized:

$$W^{k,1}_{C_i} = \frac{W^{k,1}_{C_i}}{\sum\limits_k W^{k,1}_{C_i}} \tag{3.28}$$

In this way, we have created two types of particles: the edges (for 2-node cliques), and the triangles (for 3-node cliques). We illustrated an initial set of particles in Figure 3.14.

Computing messages

Having computed initial particles from the beliefs, we can compute the particles from the messages. According to equation (3.23), we first need to marginalize the belief from the previous iteration, then divide it by the incoming message from the previous iteration. Since all node particles within the clique have one common weight (e.g., $\{W^{k,m}_{C_i}, X^{k,m}_{C_i}\} = \{W^{k,m}_{C_i}, \{X^{k,m}_t, X^{k,m}_u\}\}$), we can simply pick the particles of

[3]We implicitly assumed that $q^1_{C_i}(x_{C_i}) = 0$ if the state of one of the clique nodes is out of the deployment area.

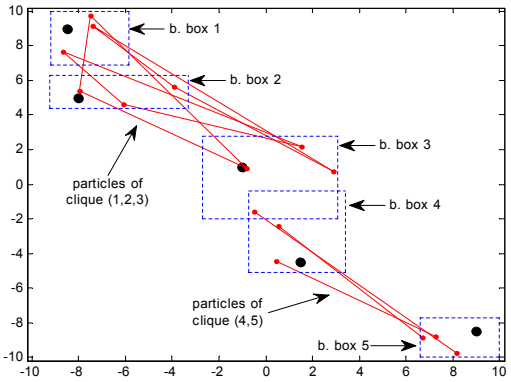


Figure 3.14: Illustration of initial particles from 2-node and 3-node cliques (for clarity, we show only 3 particles). Note that all the particles are originally high-dimensional (6D or 4D), but they can be shown in 2D space thanks to the distance constraint. The true node positions are marked with black circles.

separator nodes (from clique that sends the message), and compute the weight as reminder of (3.23). The separators sets can include one or two nodes, so there exist 1-node and 2-node messages. Therefore, the weighted particles of the 2-node message from $C_i(t,u,v)$ to $C_j(t,u,r)$, at iteration $m+1$, are given by:

$$X_{S_{ij}}^{k,m+1} = \{X_t^{k,m}, X_u^{k,m}\} \tag{3.29}$$

$$W_{S_{ij}}^{k,m+1} = \frac{W_{C_i}^{k,m}}{m_{ji}^m(X_t^{k,m}, X_u^{k,m})} \tag{3.30}$$

The 1-node messages can be found in analog way. As we can see, we need an approximation of the parametric form of the message m_{ji}^m (e.g., its KDE), so we estimate it (as for NBP) using a spherically symmetric Gaussian kernel [48,104]. The bandwidth, parameter which controls the smoothness of KDE, can be found using "rule of thumb" [48], or some advanced method (e.g., [13]). For 2-node message, it is very expensive to estimate the parametric form directly from high-dimensional (4D) particles. However, thanks to the dependency between the nodes within the message (the noisy distance), we can reduce the dimension of the message by:

$$m_{ji}^m(x_t, x_u) = m_{ji}^m(x_t)\psi_{tu}(x_t, x_u)\psi_u(x_u) \tag{3.31}$$

Note that in PJT (in contrast to JT), there is always an observed distance between each pair of the nodes within the clique (i.e., no additional "virtual" edges added by triangulation). Thus, it is sufficient to transmit the particles over one node, and upon receiving, shift them in a random direction for the observed distance.

Moreover, the messages from any anchor a to any neighboring unknown node t, are simply given by the parametric form:

$$m_{at}(x_t) = \psi_{at}(x_a^*, x_t) \tag{3.32}$$

where we assumed that position of the anchor node is perfectly known (i.e., defined by Delta Dirac function). However, if anchors' positions are uncertain (e.g., as in [108]), the message can be computed in the same way as the messages from the unknown nodes.

Computing beliefs

According to (3.24), the belief of clique i is a product of its clique potential and all the messages coming into the clique. Before drawing the particles, we need to solve two problems: i) the messages include information about different nodes within the clique, and ii) it is intractable to draw the particles from the product.

The first problem can be solved by filling the message with information about nodes which appear in destination clique, but not in the message. For example, for the message $m_{ij}^{m+1}(x_t, x_u)$, from $C_i(t, u, v)$ to $C_j(t, u, r)$, we can form the *joint message*:

$$M_{ij}^{m+1}(x_t, x_u, x_r) = m_{ij}^{m+1}(x_t, x_u)\psi_{tr}(x_t, x_r)\psi_{ur}(x_u, x_r)\psi_r(x_r) \tag{3.33}$$

Taking equations (3.31), (3.21), and (3.22) into account, the joint message can be always written as:

$$M_{ij}^{m+1}(x_{C_j}) = m_{ij}^{m+1}(x_t)\psi_{C_j}(x_{C_j}) \tag{3.34}$$

where node t must be in appropriate separator set ($t \in S_{ij}$), and if $|S_{ij}| > 1$, we can pick one node randomly. Thanks to the particles from the standard messages, we already have few (one or two) node particles from each joint message. The remained node particles can be drawn by shifting given node particles in a random direction for an amount which represents the observed distance, and by checking (only in case of 3-node clique) another distance constraint. Of course, the weights of the particles from joint messages are equal to the weights of the particles from the standard messages. However, due to the sample depletion, we *resample with replacement* [2, 32] so as to produce the particles with same weights: $\{1/N_C, X_{ij}^{k,m+1}\}$. The most of the particles, especially in the case of small noise (with small st. deviation), will be the same, which could cause very poor representation of the beliefs. Therefore, to each

of these particles, we add a small jitter ω drawn from p_v:

$$X_{ij}^{k,m+1} = X_{ij}^{k,m+1} + \omega \cdot [\cos(\theta) \;\; \sin(\theta)] \tag{3.35}$$

where θ represents the random direction ($\theta \sim Unif[0, 2\pi)$). Finally, due to the problem ii), instead of the product, we make the sum of the joint messages (i.e., using MIS [46]). Therefore, the final importance density for the belief of clique j, and corresponding particles, are respectively given by:

$$q_{C_j}^{m+1}(x_{C_j}) = \sum_{i \in G_{C_j}} M_{ij}^{m+1}(x_{C_j}) \tag{3.36}$$

$$\{W_{C_j,q}^{k,m+1}, X_{C_j,q}^{k,m+1}\Big|_{k=1}^{\left|G_{C_j}\right|}\} = \{\frac{1}{\left|G_{C_j}\right| \cdot N_c}, \bigcup_{i \in G_{C_j}} X_{ij}^{k,m+1}\} \tag{3.37}$$

We can now find the set of particles from the beliefs $\{W_{C_j}^{k,m+1}, X_{C_j}^{k,m+1}\}$ ($k = 1, ..., N_C$):

$$X_{C_j}^{k,m+1} = choose(X_{C_j,q}^{k,m+1}\Big|_{k=1}^{\left|G_{C_j}\right|}) \tag{3.38}$$

$$W_{C_j,corr}^{k,m+1} = W_{C_j,q}^{k,m+1} \frac{\prod_{i \in G_{C_j}} m_{ij}^{m+1}(X_{S_{ij}}^{k,m+1})}{q_{C_j}^{m+1}(X_{C_j}^{k,m+1})} \tag{3.39}$$

$$W_{C_j}^{k,m+1} = W_{C_j,corr}^{k,m+1} \cdot \psi_{C_j}(X_{C_j}^{k,m+1}) \prod_{\substack{t \in C_j \\ a \in G_{C_j}}} m_{at}(X_t^{k,m+1}) \tag{3.40}$$

where $W_{C_j,corr}^{k,m+1}$ is correction of the weights due to the MIS, $X_t^{k,m+1}$ particle from the node t, m_{at} is the message from the anchor node a to unknown node t, and the function *choose* chooses randomly one particle from $\left|G_{C_j}\right|$.

As a convergence parameter, we can again use approximated KL divergence between beliefs in two consecutive iterations, which is given by:

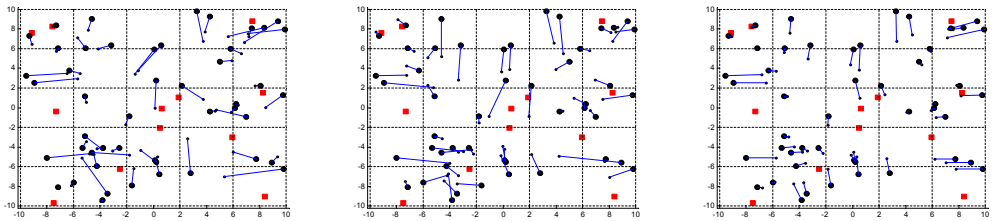
$$KL_j^{m+1} = \sum_k W_{C_j}^{k,m+1} \log \frac{W_{C_j}^{k,m+1}}{M_j^m(X_{C_j}^{k,m+1})} \tag{3.41}$$

where we used the approximation $M_j^{m+1}(X_{C_j}^{k,m+1}) = W_{C_j}^{k,m+1}$. The algorithm stops when KL_j^{m+1} (for all j) drops below the predefined threshold. Since the number of iterations is not very sensitive parameter as for NBP), we can also predefine it.

The final estimate of each node within the cliques, is given as the mean of the particles from the belief in last iteration. Since the most of the nodes appear in more than one clique, we can simply average multiple estimates or use just one of

Algorithm 5 NGBP-PJT method for localization

for all *unknown nodes* **do**
    Obtain distance estimates to all neighbors
    Construct bounded box
    Set all parameters to the initial values
end for
Form PJT using Alg. 3 and Alg. 4
for all *cliques* **do**
    Draw initial particles from the importance density function (3.25) or (3.26)
    for $m = 1 : N_{iter}$ **do**
        Compute particles for outgoing messages via (3.29)-(3.30)
        Compute (eventual) messages from anchors via (3.32)
        Compute KDE of the messages
        Draw particles from the joint messages (3.34)
        Resample with replacement
        Add small jitter to all particles via (3.35)
        Compute particles from the beliefs via (3.36)-(3.40)
    end for
end for
Compute final location estimates



(a) (b) (c)

Figure 3.15: Illustration of results for the 60-node network: (a) NBP, (b) NBP-TG, and (c) NGBP-PJT. The anchors are marked with red squares, the unknowns with black circles, and the estimated locations with black dots.

them. We summarize the NGBP-PJT algorithm in Alg. 5.

Finally, it is worth noting that a special case of NGBP-PJT method is the NBP method based on TG (NBP-TG) assuming that TG has only the pairs of the nodes as cliques. NBP-TG is very interesting by-product since it runs on the same graph as NGBP-PJT, which makes this method cheaper than NBP. It also helps to understand how much removed edges from the original graph change the performance of the method (see following section).

3.5.5 Simulation results

We assume that there are $N_a + N_u = 60$ nodes in 20m x 20m area. The minimum number of anchors (which are deployed near the edges) is 4. This, usually realistic, constraint helps the unknown nodes near the edges which suffer from low connectivity. The rest of the anchors and the unknowns are randomly deployed within the area. The number of iterations is set to $N_{iter} = 3$, which means that each node/clique will receive all the information available 3-hop away from itself. The transmission radius is set to $R = 8$m. Simulations are performed using $N_a = 6$ and $N_a = 12$ anchors. We assume that the distance is obtained from the RSS measurements using log-normal model, since this is usually the worst case scenario [76]. Thus, we choose σ_{dB} =5dB as standard deviation of RSS (i.e., the parameters[4] of the log-normal distribution are $\mu = log(d)$ and $\sigma = \sigma_{dB}/10n_p = 0.25$, where $n_p = 2$ is the path-loss exponent[5]). Previous parameters are same both for NBP, NBP-TG, and NGBP-PJT. However, the number of particles, is set to 100 (for NBP), 290 (for NBP-TG) and 210 (for NGBP-PJT), so as to make nearly the same computational time for all three methods (see Table 3.1). For the KDE of the messages, the bandwidth is found using "rule of thumb", which is the simplest option. The following simulation results represent the average over 20 Monte Carlo runs. Note that all defined parameters are valid only if not otherwise stated in the following text.

Comparison of accuracy and convergence

Using the defined scenario, we compare the accuracy and the convergence of NBP, NBP-TG and NBP-PJT algorithms. The error is defined as Euclidean distance between the true and estimated location. First, we illustrate the results of these methods in Figure 3.15. We can see that NBP-PJT method significantly outperforms both NBP and NBP-TG methods, and also that NBP slightly outperforms NBP-TG. Moreover, for the randomly chosen node, we illustrate its initial and final belief in Figure 3.16. Obviously, the initial beliefs of NBP and NBP-TG represent nearly uniform distribution within its bounded box, but the initial belief of NGBP-PJT (which is also within its bounded box) is not uniform due to the distance constraints within appropriate cliques (see (3.25)). Thus, the initial belief of NGBP-PJT is more informative than the belief of NBP. We can also see that final NBP belief is tighter (i.e., more informative), but this is because of the overconfidence caused

[4]Note that these values do not represent the mean value and the standard deviation of the distance. They are respectively given by: $\mu_d = e^{\mu+\sigma^2/2}, \sigma_d = \mu_d\sqrt{e^{\sigma^2}-1}$. Consequently, these parameters are distance dependent.

[5]Typical values for n_p are between 2 and 6 [84]. For the distance estimation, the minimum value is the worst case.

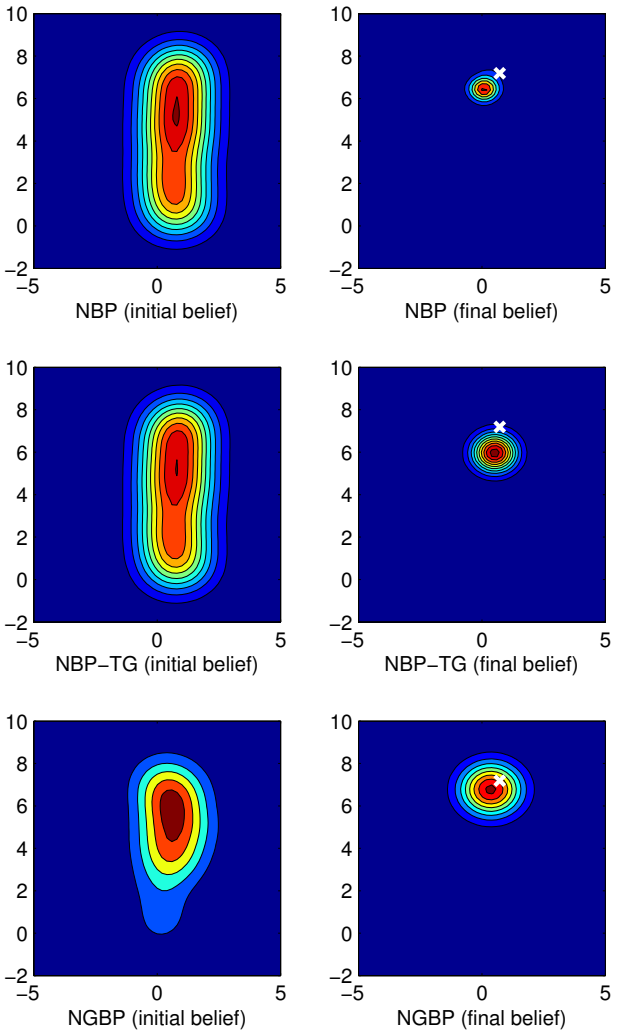


Figure 3.16: Comparison of NBP, NBP-TG and NGBP-PJT beliefs in the first and last iteration. True position of the node is marked with X.

by loopy networks. That means that the true position of the node can be placed in the area with probability close to zero (as shown in Figure 3.16). On the other hand, NBP-TG and NGBP-PJT are less informative, but more trustful. NBP-TG is still based on loopy graph, so slightly overconfident comparing with NGBP-PJT. In order to obtain more precise conclusion about accuracy, we also consider cumulative distribution function (CDF) of the error in position. We can see in Figure 3.17 that NGBP-PJT outperforms all other methods in terms of maximum, minimum, and median error (and also any other percentile).

Furthermore, we provide the analysis of the root-mean-sqaure (RMS) error with respect to transmission radius. According to Figure 3.18, the NGBP-PJT significantly (5-10%) outperforms NBP and NBP-TG, for all R and both values of N_a. It is also worth noting that the number of anchors significantly affects accuracy.

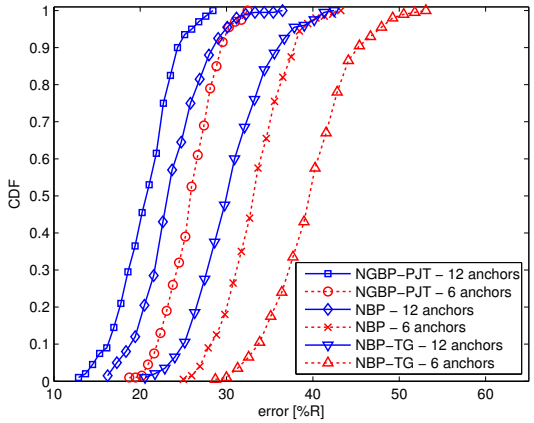


Figure 3.17: CDF of the RMS error in position

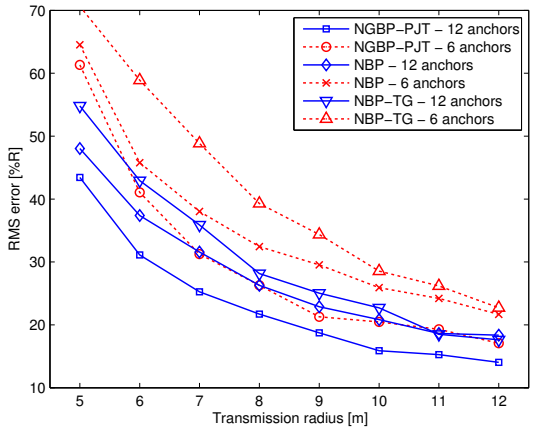


Figure 3.18: The effect of transmission radius on RMSE in position

For instance, NGBP-PJT with 6 anchors performs similar as NBP with 12 anchors. Therefore, given nearly the same accuracy, one can decrease the equipment cost by removing 6 anchors (which are usually very expensive). It is also interesting to note the performance difference between NGBP-PJT and NBP-TG since they are based on same graph.

Regarding the convergence, we can see in Figure 3.19, that all algorithms converge sufficiently after second iteration. Note that this is expected since we set $R = 8$m, so almost all information is maximum 2-hop away from each clique. Anyway, we chose one more iteration because we consider also smaller values of R ($R_{min} = 5m$). Finally, we can see that all algorithms, especially NBP-TG, can not perfectly converge (i.e., $KL \to 0$) after reasonable number of iterations. This is, of course, caused by the existence of loops (for NBP and NBP-TG), and missing edges in TG (for NBP-TG and NGBP-PJT).

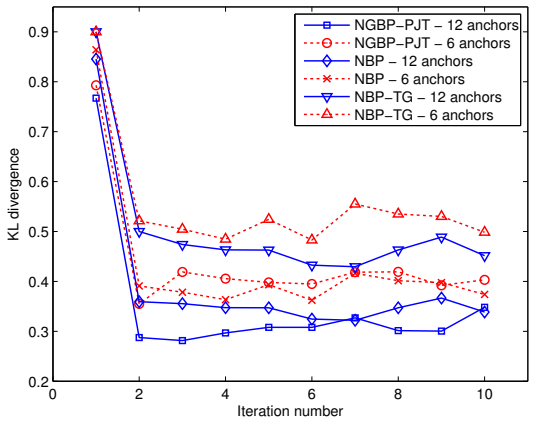


Figure 3.19: Comparison of KL divergence in each iteration

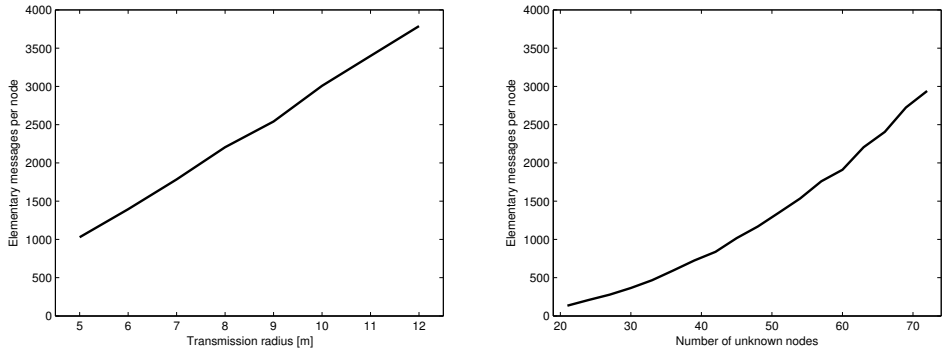
Table 3.1: Comparison of the computational cost (measured in MFlops)

R	NBP	NBP-TG	NGBP-PJT	PJT
5m	0.56	0.56	0.56	0.007
8m	0.71	0.59	0.67	0.011
12m	0.82	0.62	0.73	0.013

Comparison of computational and communication cost

As we already mentioned, we set the same computational cost for $R = 5$m by choosing appropriate number of particles for all three methods. It was not possible to set the same cost for all methods ($R > 5$m) because the cost is more sensitive to R in case of NBP. On other hand, NGBP-PJT and NBP-TG costs are less sensitive to R due to the nearly same number of edges with respect to R, in formed TG. We provide the average cost per node for different values of R in Table 3.1. We can see that the cost of NGBP-PJT is the same or less for all considered values of R. We can also see that complexity of the PJT formation is negligible comparing with full algorithms.

Regarding communication cost, which is very important for the battery life of the wireless devices, we count elementary messages (i.e, scalar values). We will consider the effect of transmission radius and number of unknowns, since their variations obviously affect the cost. First, we analyse the cost of PJT formation (Alg. 1 and Alg. 2). As we can see in Figure 3.20, it is a linear function of transmission radius, and a quadratic function of number of unknowns. Second, we analyse the cost of all considered algorithms w.r.t. R for 2 different number of unknowns. According to Figure 3.21, we can conclude the following:



(a) (b)

Figure 3.20: Communication cost for PJT formation w.r.t. (a) transmission radius, (b) number of nodes.

- NGBP-PJT significantly outperforms NBP and NBP-TG methods, for reasonable number of unknowns.
- Comparing with NBP, the improvement of NGBP-PJT is increasing as transmission radius increasing. This is achieved thanks to the TG.
- The cost of NBP-TG is slightly less than NGBP-PJT due to the redundancy in PJT graph (i.e., when the same node appears in more than one clique).
- Increasing the number of unknowns will decrease the benefit of NGBP-PJT. This is caused by quadratic dependency of PJT formation w.r.t. number of unknowns. Using results from Figure 3.20b and Figure 3.21, we estimate that NGBP-PJT will reach the same cost as NBP, for 140 unknown nodes.

Finally, we can conclude that the proposed NGBP-PJT method is cheaper for reasonably-sized networks. However, it can also be cheap for very large-scale networks if the network is divided into regions, and one PJT created for each of them.

3.6 Nonparametric belief propagation based on spanning trees (NBP-ST)

The GBP-based algorithms, even with approximations, are still very complex for large-scale ad-hoc/sensor networks. Moreover, the connectivity in these networks is very high, which introduces computational and communication burdens for low-power applications. Therefore, we propose a technique to simplify the algorithm by breaking the loops using NBP-ST [91] created by a BFS method [7] (which is already used, in different form, for TG formation in NGBP-PJT).

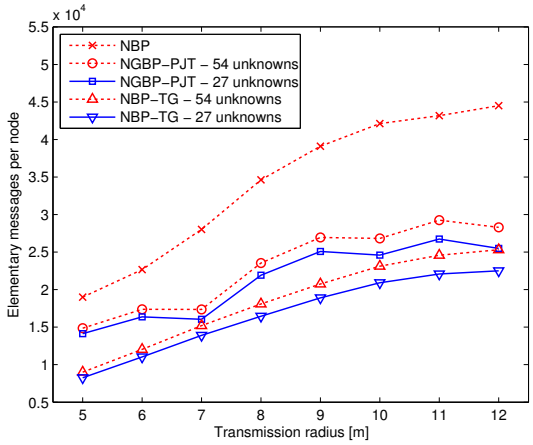


Figure 3.21: The effect of transmission radius on communication cost (for 2 different number of unknowns). NBP cost is the same for any number of unknowns, so we plot just one curve.

Algorithm 6 ST formation using BFS method

Input: list of nodes Q and root node $root$
Set current root: $r \leftarrow root$
while Q is not empty **do**
    for all nodes $t \in G_r$ **do**
        if $t \in Q$ **then**
            Remove t from Q
            Insert t in Q_r
            Insert d_{rt} in S
        end if
    end for
    Set current root: $r \leftarrow$ first unused node from Q_r
end while
Output: spanning tree $\{Q, S\}$

3.6.1 ST formation

A ST is an acyclic subgraph that connects all the nodes of the original graph. The optimal method for ST formation for unweighted graphs is using a BFS, which is already described in Section 3.5.3. Here we provide the detailed pseudocode in Alg. 6.

In the case of NBP localization, we exclude all the anchors from the BFS algorithm since they do not form the loops in the graph (they just send, and never receive the messages). Since the ST is very coarse approximation of the original graph, we need to create at least two of them. A graph generally has a large number of STs, but we can choose 2 (or more) in a semi-random way. The first root node we choose randomly from the set of all unknown nodes. In order to maximize the difference between two STs, the second root node has to be as far as possible from

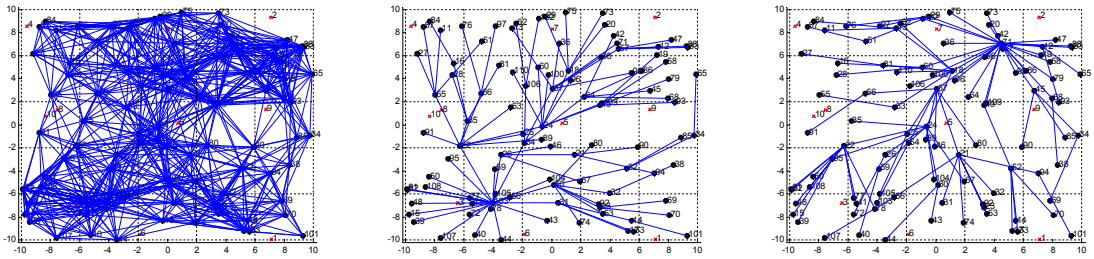
the first root node. Thus, it should be one of the leaf nodes. If we want to form more STs, the analog constraint will be used. Note that, using BFS, it is not possible to form two STs with completely different edges and that usually some of the edges will be out of both STs. If we want to include all the edges, we have to add more STs but it is not necessary since it will likely provide us redundant information. It is especially the case in the WSNs with high connectivity.

Since NBP is naturally distributed method, the proposed BFS method has to be done in a distributed way. This can be simply done if each unknown node initially broadcasts its *ID* to all the neighbors, which will continue to broadcast to others, and so on, until each unknown node has a list of all unknown nodes in the graph. One node (e.g. with the lowest *ID*) has to be assigned to choose the root node from the list and give him permission (by multihop broadcasting) to start the BFS algorithm. Then, the chosen root node has all initial data to start the BFS algorithm, and, when it is necessary, has only to broadcast all data (i.e. variables from Alg. 6) to all its neighbors. The last visited node will have available the final ST.

Algorithm 7 NBP-ST method for localization

1: **for all** nodes **do**
2: Take sensing actions
3: Set all parameters to the initial values
4: Broadcast own and all received *IDs* and listen for other sensor broadcasts (until receive all *IDs*)
5: **end for**
6: Set a list of nodes for BFS (excluding anchors): Q
7: Choose randomly root node from the list Q: *root*
8: **for all** *spanning trees* **do**
9: Run BFS (Alg. 6)
10: Run NBP on defined ST
11: Choose root node as far as possible from the previous roots
12: **end for**
13: Fuse all beliefs into one and compute location estimates

Finally, NBP-ST algorithm represents two (or more) independent runnings of the NBP algorithm based on formed STs. Each running will provide weighted particles of the node beliefs computed by (3.7). The simplest way to fuse these beliefs is to draw particles from the product of the beliefs from different STs. This can be done using MIS (see Section 3.3.2). The collection of weighted particles from all STs represents our final output, from which we can easily extract any estimate that we need (e.g., mean value or variance of the location estimate). The pseudocode in Alg. 7 illustrates the NBP-ST method.



(a) (b) (c)

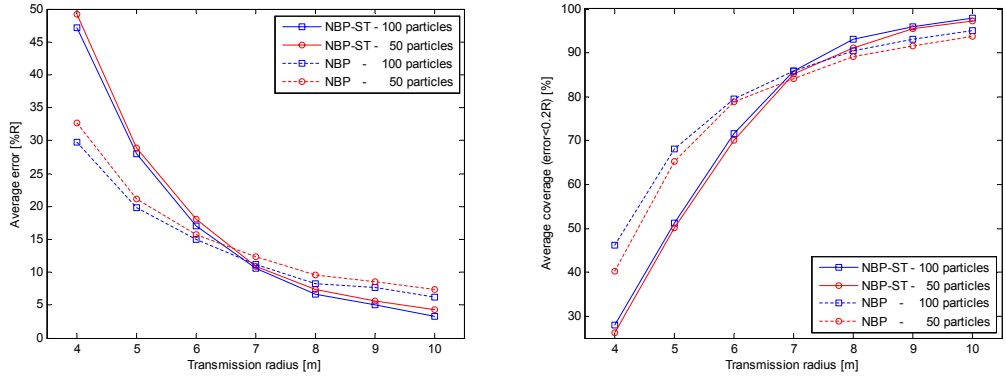
Figure 3.22: (a) Original network with 100 unknown nodes (black dots) and 10 anchors (red asterisks) for $R = 6m$, and (b), (c) Two corresponding STs. The roots are node 103 and node 71, respectively.

3.6.2 Simulation results

To illustrate the performance of this method, we conducted several simulations. We assume that there are 100 unknown and 10 anchor nodes randomly deployed in 20m x 20m area. Since the unknown nodes near the edges of the deployment area suffer from low connectivity, we make an exception for four anchors, which are randomly deployed within four square-shaped areas of 4m x 4m near the edges. The standard deviation of the Gaussian noise on the distance estimate is set to $\sigma = 0.3$m and the number of iteration is set to $N_{iter} = 3$. All simulations are done for $N = 50$ and $N = 100$ particles with respect to the transmission radius (R = 4m - 10m). Each point in the simulations represents the average over 20 Monte Carlo trials.

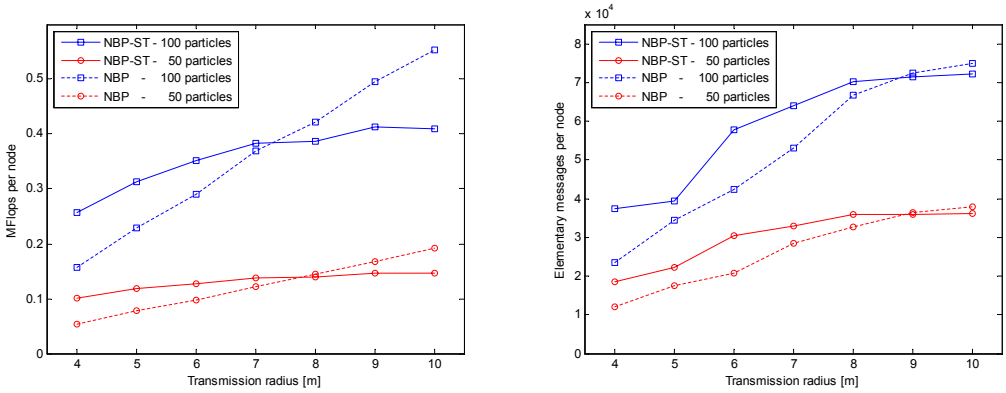
Using the defined scenario, we compared NBP and NBP-ST algorithms. For NBP-ST, we used 2 STs. The original network and 2 STs created by BFS are illustrated in Figure 3.22. Regarding accuracy and coverage in Figure 3.23, NBP-ST performs better than NBP for $R >$ 7m, approximately. Obviously, for these values of R there is a large number of loops in the network which decreases the performance of the NBP method. For lower values of R, we could expect that NBP-ST performs with higher (or same) accuracy, but we cannot forget that, by using only 2 STs, we do not include all information (i.e., removed edges) that we have. Thus, the NBP outperforms NBP-ST in this case.

Regarding the computational/communication cost (Figure 3.24), NBP-ST performs better than NBP for $R >$ 8m and $R >$ 9m, respectively. In order to explain this result, we should recall two main things we have taken into account: i) removing the edges in order to form the STs, and ii) running NBP two times in these STs. The former decreases the computational/communication cost, but the latter one increases it. Therefore, in the low-connected networks the second operation predominates, but in the highly-connected networks the first one predominates. The



(a) (b)

Figure 3.23: Comparison of (a) accuracy and (b) coverage



(a) (b)

Figure 3.24: Comparison of (a) computational and (b) communication cost

additional contribution is that (for high transmission radius), the computational and the communication costs are nearly constant. This feature provides us more precise information about battery life of the sensors. The final conclusion is that the NBP-ST algorithm performs better than NBP in all terms, for $R > R_{min}$. In our case $R_{min} = 9$m, but this parameter depends on the density in the WSN (i.e. average node degree).

3.7 Uniformly-reweighted nonparametric belief propagation (URW-NBP)

For the previous proposed methods (NBP-ST, NGBP-JT and NGBP-PJT), we need to make some kind of graph transformations before applying message passing

method. This usually reduces the robustness of the whole algorithm, since the failure of just one node can make significant effect on the localization performance. In addition, it is necessary to synchronize whole network, which is sometimes not possible. Since standard NBP is robust to node failures, and capable to run in asynchronous networks [24], our goal is to find an improved method with the same features. Therefore, we propose a method based on tree-reweighted BP (TRW-BP) proposed in [110].

In the standard TRW-BP algorithm the belief at a node t is proportional to the product of the local evidence at that node $\psi_t(x_t)$, and all *reweighted* messages coming into node t:

$$M_t(x_t) \propto \psi_t(x_t) \prod_{u \in G_t} m_{ut}(x_t)^{\rho_{ut}}, \tag{3.42}$$

where x_t is a state of node t, $\rho_{tu} = \rho_{ut}$ is the appearance probability of the edge (t, u), and G_t denotes the neighbors of node t. The messages are determined by the message update rule:

$$m_{ut}(x_t) \propto \int \psi_u(x_u)\psi_{tu}(x_t, x_u)^{1/\rho_{tu}} \prod_{k \in G_u \setminus t} \frac{m_{ku}(x_u)^{\rho_{ku}}}{m_{tu}(x_u)^{1-\rho_{tu}}} dx_u, \tag{3.43}$$

where $\psi_{tu}(x_t, x_u)$ is the pairwise potential between nodes t and u. On the right-hand side, there is a product over all reweighted messages going into node u except for the one coming from node t. The update-rule (3.43) is carried out over the network. Upon convergence, the beliefs are computed through (3.42). As in the case of NBP, it is more convenient to compute the beliefs at every iteration i. This leads to an equivalent form of TRW-BP: by replacing (3.42) in (3.43), we find that the belief equations and message-update rule of TRW-BP are, respectively, given by:

$$M_t^i(x_t) \propto \psi_t(x_t) \prod_{u \in G_t} m_{ut}^i(x_t)^{\rho_{ut}} \tag{3.44}$$

$$m_{ut}^i(x_t) \propto \int \psi_{ut}(x_t, x_u)^{1/\rho_{ut}} \frac{M_u^{i-1}(x_u)}{m_{tu}^{i-1}(x_u)} dx_u. \tag{3.45}$$

We can now apply TRW-BP to the localization problem. In the first iteration of this algorithm it is necessary to initialize $m_{ut}^1 = 1$ and $M_t^1 = p_t$ (i.e., information from anchors, if any) for all u, t, and then repeat computation using (3.44) and (3.45) until convergence or a preset number of iterations is attained. In a practical implementation, we have to use nonparametric version of TRW-BP (TRW-NBP). Hence, the beliefs and message update equations, (3.44) and (3.45), are performed using particle-based approximations. Since this approximation is done in the same way as for NBP, see Section 3.3 (and also [46, 88]) for more details.

3.7.1 Edge appearance probabilities

We will now describe how valid values for ρ_{tu} can be found. Given a graph G, let S be the set of all STs T over G. Let $\vec{\rho}$ be a distribution over all STs, i.e., a vector of non-negative numbers such that

$$\vec{\rho} \stackrel{\Delta}{=} \{\rho(T),\ T \in S \mid \rho(T) \geq 0\ ,\ \textstyle\sum_{T \in S} \rho(T) = 1\}. \tag{3.46}$$

Observe that there are many such distributions. For a given $\vec{\rho}$ and a given (undirected) edge (t,u), $\rho_{tu} = P_{\vec{\rho}}\{(t,u) \in T\}$, i.e., ρ_{tu} is the probability that the edge (t,u) appears in a ST T chosen randomly under $\vec{\rho}$. Thus, ρ_{tu} represents *edge appearance probability* of the edge (t,u). A valid collection of edge appearance probabilities must correspond to a valid distribution over STs. For instance, $\rho_{tu} = 1$ for all edges, is not a valid collection of edge appearance probabilities, unless the graph G is a tree.

Finding a valid collection ρ_{tu} is difficult since there is a large number of STs even in small graphs. For example, in 4-node clique there are 16 different STs (Figure 3.25), and each edge appears exactly in 8 of them. Observe that if $\vec{\rho}$ is uniform over all STs, then $\rho_{tu} = 0.5$ for every edge. Discovering all STs, choosing a good $\vec{\rho}$, and then computing all ρ_{tu} would require significant network overhead, even for small networks. In [56], an alternative option is described, based on searching for trees (not necessarily STs) in G. In any case, determining a valid collection ρ_{tu} requires a procedure similar to routing, which we prefer to avoid in order to make this method more robust to failures.

We note that the choice $\rho_{tu} = 1$ for all edges corresponds to standard BP. In TRW-BP on graphs with cycles, it is easy to see that $\rho_{tu} \leq 1$ for all edges. Hence, by removing the restriction of valid ρ_{tu} and making ρ_{tu} uniform, we intuit that we can combine the benefits of NBP (distributed implementation) and of TRW-NBP (improved performance). This leads to the novel method, *uniformly reweighted* NBP (URW-NBP) [119]. We apply this method for cooperative localization [95] with the same model as for NBP, but it can be successfully applied in many more applications [79, 120].

3.7.2 Simulation results

We consider URW-NBP (with $\rho_{tu} = \rho$ for all edges) and NBP ($\rho_{tu} = 1$). The goal is to evaluate the impact of ρ on URW-NBP through Monte Carlo simulation. We will first consider a small-scale network with 4 nodes, for which we can compute the true marginal posterior PDFs. From this network, we will draw some important

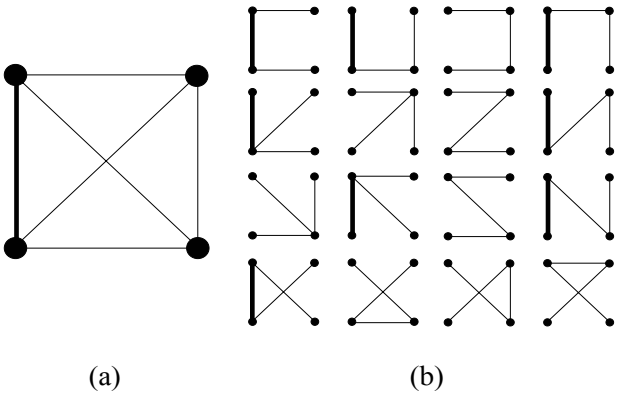


Figure 3.25: (a) 4-node clique, and (b) 16 STs. Each edge (e.g., bolded edge) appears exactly in 8 out of 16 STs, so $\rho = 0.5$ for each edge, under a uniform distribution over the STs.

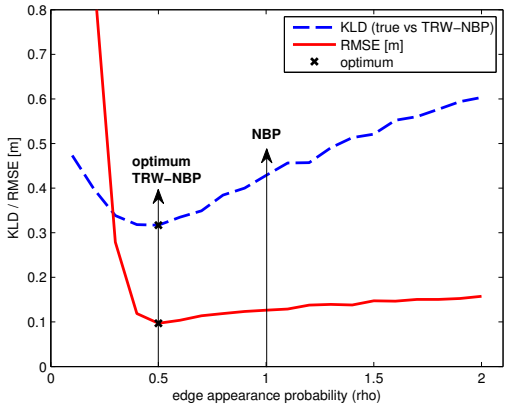


Figure 3.26: Optimum ρ estimation in 4-node network.

conclusions necessary for larger networks. Then, we determine the optimal ρ, with respect to transmission radius, in grid and random topologies. Due to the high computational cost of learning the optimal ρ in a 2D space, we will mostly focus on 1D localization. We use the following parameters: standard deviation of Gaussian noise is $\sigma = 0.3$m, $N = 200$ particles per message, and $N_{iter} = 8$ iterations. Finally, all results represent the average over 200 Monte Carlo runs.

A 4-node clique

We consider fully-connected network with 4 targets in 1D space (see Figure 3.25a for 2D case). In addition, there are 4 anchor nodes (not depicted), each of them connected exactly to one target. Our goal is to estimate the true belief, TRW-NBP beliefs and estimated locations (note that URW-NBP and TRW-NBP are equivalent for this case). The latter are given by the MMSE estimate from the belief. We run

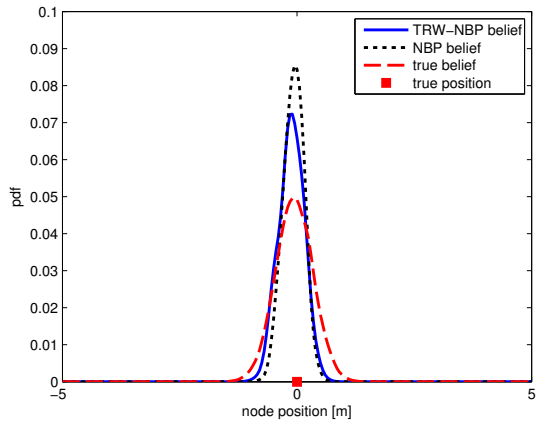


Figure 3.27: NBP, TRW-NBP ($\rho = 0.5$), and true belief for the one of the target nodes ($x = 0$m).

TRW-NBP for different values of ρ and, for each result, we compute KLD between true and TRW-NBP beliefs, and RMSE of estimated locations, all shown in Figure 3.26. According to Figure 3.26, we can make the following conclusions:

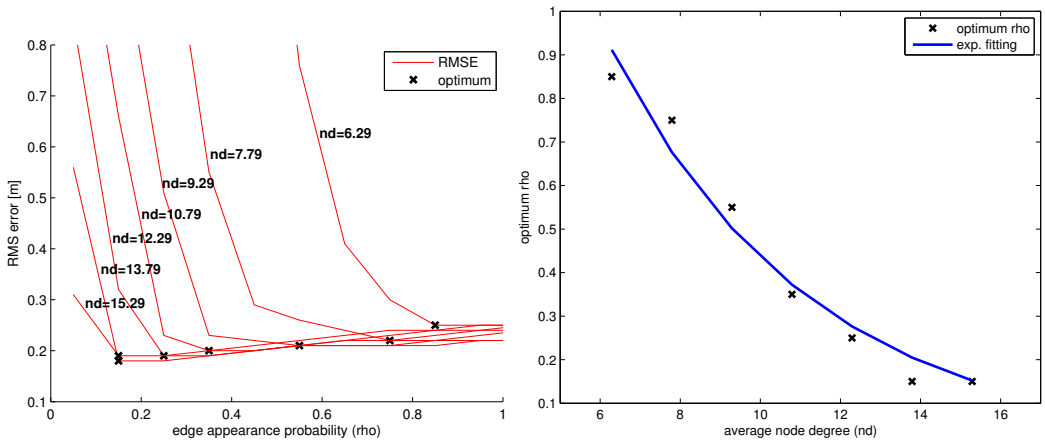
- Both RMSE and KLD reach the minimum for the same $\rho < 1$. That means that it is sufficient to use only RMSE for learning the optimal ρ in larger networks, where the computation of true beliefs (necessary for computing KLD) is intractable.

- The optimal ρ (ρ_{opt}) is 0.5, which is the same as the theoretical value (Figure 3.25b), under a uniform distribution over STs. NBP ($\rho = 1$) performs worse than optimum TRW-NBP in terms of both KLD and RMSE. Note that TRW-NBP belief is still an approximation, so the true belief is still the most accurate representation of the location estimate. For a comparison between the three different beliefs, see Figure 3.27.

- A wide range of ρ (in our example, 0.4-1) provides better performance than NBP in terms of both KLD and RMSE. That means that we can even use a coarse approximation of ρ_{opt}.

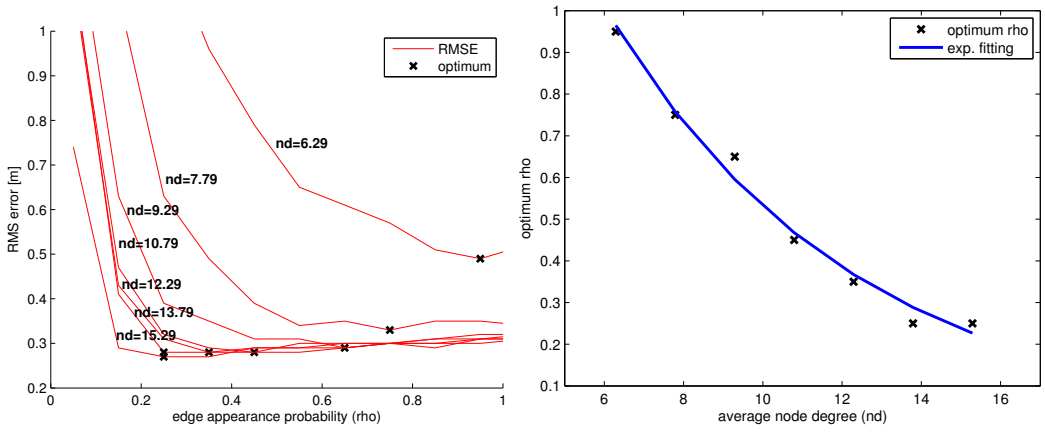
- The RMSE is rather insensitive to ρ, for $\rho > \rho_{\text{opt}}$. Hence, care needs to be taken when interpreting RMSE figures as a function of ρ, as the effect on KLD may be much more pronounced.

Taking these conclusions into account, we now move on larger networks.



(a) (b)

Figure 3.28: Grid topology: (a) RMSE for different transmission radius, (b) Empirical model for optimal ρ.



(a) (b)

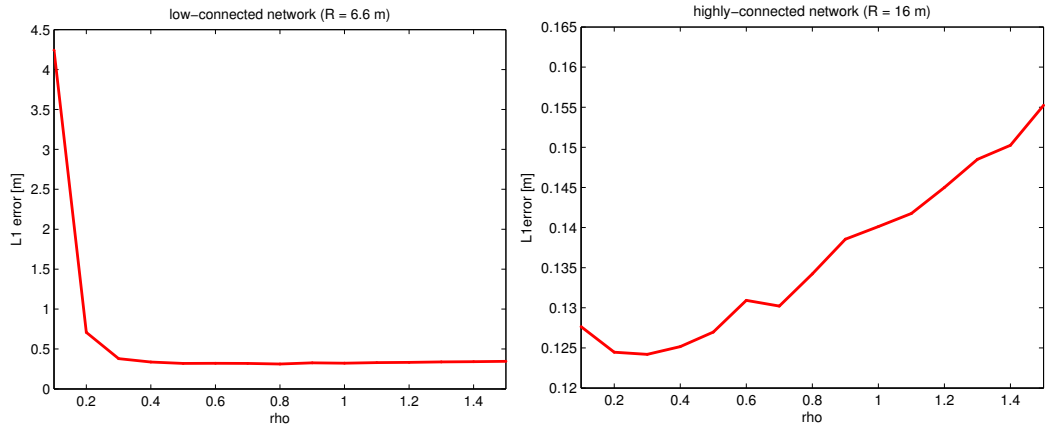
Figure 3.29: Random topology: (a) RMSE for different transmission radius, (b) Empirical model for optimal ρ.

Grid and random topology networks

We consider a network with 25 target nodes and 4 anchors in a 20m wide deployment area. We consider different values of the communication range[6] R, and the edge appearance probability ρ.

For the grid topology (where the distance between neighboring nodes is 0.6 m), Figure 3.28a shows the RMSE as a function of ρ, with parameter n_d. We mark the optimal ρ, for each distinct value of n_d. This allows us to plot ρ_{opt} as a function

[6]The values of R are chosen so as to provide the same average node degree (n_d) both for grid and random topology.



(a) (b)

Figure 3.30: Comparison of the error for: (a) $R = 6.6$m, and (b) $R = 16$m.

of n_d (see Figure 3.28b). We observe that that ρ_{opt} decreases nearly exponentially with n_d. Hence, we fit $\rho_{\text{opt}}(n_d)$ as

$$\rho_{\text{opt}}(n_d) = \rho_0 \cdot e^{-k_\rho n_d}, \tag{3.47}$$

where parameters $\rho_0 = 3.187$ and $k_\rho = 0.199$ are found using least-square fitting. We did the same test for random topology (Figure 3.29), and obtained: $\rho_0 = 2.656$ and $k_\rho = 0.161$. Note that for random topology, it is harder to obtain sufficient statistics (Figure 3.29a), so the fitting is less confident compared with the grid topology. We conclude the following:

- The difference between coefficients for random and grid topology is small, which means that the value of ρ_{opt} depends more on the average node degree than the particular network configuration.
- Though tempting to state that choosing $\rho = 1$ will lead to similar performance as $\rho = \rho_{\text{opt}}$, due to the almost flat curves for $\rho > \rho_{\text{opt}}$, this statement is not true when the performance is measured in terms of KLD (see Figure 3.26).

As an aside, when n_d becomes very small, the fitted value for ρ_{opt} can be larger than 1. This is merely a side-effect of the fitting. In practice, when $\rho_{\text{opt}} > 1$, one should set $\rho_{\text{opt}} = 1$. It is also important to note that if we do not know n_d in advance, we can n_d can easily and quickly find it using average consensus algorithm [72]. Even in that case, the computational/communication cost will be nearly the same as for NBP.

We also performed simulations for 2D space (random topology), but due to the

computational problems, only for 2 representative values of R. In this case, we used $N = 500$ particles, and $N_{iter} = 10$ iterations. We observe in Figure 3.30a that for low connectivity ($R = 6.6$m), the error is relatively insensitive to ρ for any $\rho > 0.4$. When the connectivity is increased ($R = 16$m), the best value of ρ is 0.3, while $\rho = 1$ induces around 20% additional error (Figure 3.30b). Therefore, we can conclude that the behaviour is similar as in 1D space.

3.8 Summary

In this chapter, we proposed several novel methods for cooperative localization based on nonparametric message passing methods. We provided detailed description of standard BP and NBP methods, and also proposed NBBP method which is capable to archive better performance than NBP with fewer particles. However, since these methods predict beliefs that are inaccurate in loopy networks, we proposed four solutions: NGBP-JT, NGBP-PJT, NBP-ST, and URW-NBP. The NGBP-JT method, which provides accurate beliefs in loopy networks, has acceptable complexity only in small-scale sensor networks. NGBP-PJT can significantly outperform NBP, but it is not fully scalable. NBP-ST can slightly outperform NBP method in highly-connected networks, and it is computationally feasible in large-scale ad-hoc/sensor networks. However, the problem of all these methods is that some kind of graph transformations is necessary before applying message passing method, which decreases their robustness to failures. Therefore, we also proposed URW-NBP, which is capable to slightly outperform NBP while keeping NBP's robustness to failures.

Chapter 4

Cooperative mobile network localization and tracking

4.1 Introduction

This chapter addresses two important problems: i) cooperative localization in mobile networks, and ii) cooperative tracking of the passive object. For both problems, we apply a variants of nonparametric message passing techniques. For the cooperative localization in mobile networks, we extend NBP described in Chapter 3. In contrast to previous methods, we send an optional message from the future to present using only 1-leg smoothing, and solve two important problems of the standard NBP method: decrease the communication cost, and increase the efficiency of the sampling techniques. Our new low-cost protocol, which requires communication of the beliefs (instead of the messages), which are approximated with the Gaussian mixture of very few components, is almost as accurate as the transmission of the particles. This protocol also applies censoring, i.e., only informative data have been transmitted. Regarding sampling techniques, we improve standard MIS by adding uniformly distributed particles, which makes NBP robust in the case of outliers. Moreover, we apply two sampling techniques within NBP, which are based on *population Monte Carlo* (PMC), and *auxiliary* variable. These techniques increase the amount of information in the importance density. For the second problem, cooperative (and distributed) tracking, the goal is to track the passive object which cannot locate itself (in contrast to cooperative mobile localization). Therefore, all the nodes in the network must agree on the estimate of the target state. Since the current state-of-the-art do not use fastest consensus methods, and also most of them cannot handle all parametric and nonparametric likelihood functions, we propose novel general framework for distribute target tracking. We use distributed particle filter-

ing (DPF) based on three asynchronous belief consensus (BC) algorithms: standard belief consensus (SBC), broadcast gossip (BG), and belief propagation (BP). Since DPF can be also solved (without consensus) by exchanging the observed data, we also determine under which conditions BC-based methods are preferred.

4.2 Cooperative localization in mobile networks

Cooperative localization in mobile networks is an important problem, as the availability of location information can enable many applications [21, 76, 98], such as tracking vehicles on roadways, firefighters in building under fire, forklifts in a warehouse, animals in woods, intruder detection, search-and-rescue, etc. The scenario is similar as for static localization (see Section 2.1), but now the target nodes are mobile (i.e., attached to the objects that should be tracked). Anchor nodes are usually static, but this is not necessary if they are equipped with GPS [75].

A number of methods for cooperative localization has been proposed, but most of them are used for the localization in static networks (see [46, 78, 97, 109, 112] and Chapter 2 of this thesis). Repeating these static localization algorithms can provide the location estimates in mobile networks, but this is suboptimal due to lack of the additional information given by mobility of the sensor nodes. Some works already take this information into account, for example [8, 49, 99, 118, 122]. Moreover, the goal of most localization methods [49,109,112] is just to estimate the position of all target nodes, without the associated uncertainty. Since uncertainty of the estimate is crucial for most applications, Bayesian approach [46,78,118] can be applied, in which the goal is to estimate the posterior marginal PDF of the target's positions, given the priors, and the likelihood of the measurements. Since this approach is intractable in large networks, it is necessary to use some message-passing method [77] and also to approximate all distributions using particle-based approximation [2, 32]. One suitable framework can be NBP, which is initially proposed for static networks [46], and analysed in Chapter 3 of this thesis. A variants of this method have been already used for cooperative localization in mobile networks [99, 118]. In [118], authors propose particle-based distributed message passing method defined on factor graph. Comparing with NBP, which is defined on Markov Random Field, the main difference is capability to work with higher-order potentials. In [99], authors use NBP method for distributed tracking of the mobile robots. For this application, it is also necessary to estimate the speed of the targets (not only positions). It also takes advantage of bidirectional nature of the NBP to send the messages from the future to present (also known as *smoothing*). However, this is only possible in the case of offline postprocessing.

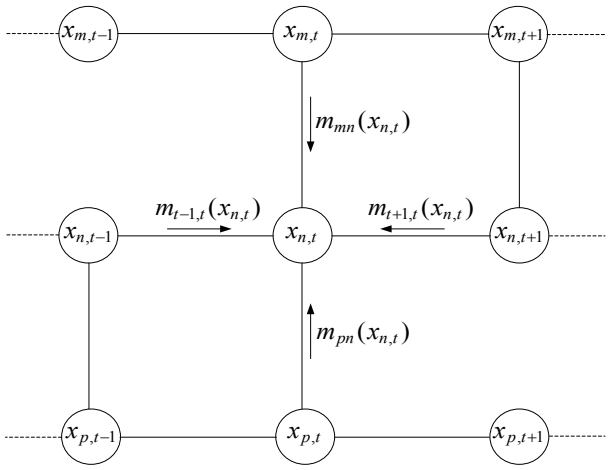


Figure 4.1: Example of graphical model for mobile positioning, which illustrates three target nodes (m, n, and p) in three consecutive time frames ($t-1$, t, and $t+1$).

In the following sections, we extend NBP method for mobile positioning. In contrast to [99, 118], we send an optional message from the future to present using only 1-leg smoothing (i.e., almost in real-time). Then, we focus on solving the communication and sampling problems of standard NBP method.

4.2.1 Extension of NBP for mobile networks

We assume that the target nodes are moving within deployment area, and that the anchor nodes are still static. Our goal is to adapt static NBP method for mobile positioning. We start with an example of the graphical model, in Figure 4.1, which illustrates three target nodes (m, n, and p) in three consecutive time frames. For instance, to locate node n at time t, we need the messages from its neighbors (m and p) as in static NBP, plus two additional messages from the past and the future. Thus, to extend static NBP, we just need to define the pairwise potential and the messages between two consecutive time frames. It is also worth noting that the connectivity between nodes can change over time.

Pairwise potential between two consecutive time frames (for target node n) $\psi_{t-1,t}(x_{n,t-1}, x_{n,t})$ (we refer to it as *kinematic potential*) represents the correlation between positions in these time frames, which depends on the kinematic of the node. There are a number of models for kinematic. If we can estimate the amplitude of the speed v_{t-1} at time $t-1$, and have distribution p_w of the process noise w (which represents a random variation of the speed due to the acceleration), the kinematic potential is given by:

$$\psi_{t-1,t}(x_{n,t-1}, x_{n,t}) = p_w(\|x_{n,t} - x_{n,t-1}\| - v_{t-1} \cdot T_S). \tag{4.1}$$

where T_s represents the sampling interval. Note that it is usually hard to measure the process noise (see Chapter 5), so Gaussian approximation is a common choice. However, since we prefer to keep non-Gaussian nature of NBP, we apply a simpler model (as in [8]) which only requires the knowledge of the maximum speed of the target V_{max}. This is usually easy to find for most applications (e.g., 5 m/s for people, 1 m/s for forklifts, 30 m/s for cars, etc.). Given V_{max}, kinematic potential of node n can be written as:

$$\psi_{t-1,t}(x_{n,t-1}, x_{n,t}) = \begin{cases} 1, & if \ \ \|x_{n,t-1} - x_{n,t}\| \leq V_{max} \cdot T_s, \\ 0, & otherwise. \end{cases} \tag{4.2}$$

Using this potential, we can predict the possible positions of node n at time t, given the estimate at time $t-1$, and vice versa for smoothing. Note that, if we can measure the dynamic of the target (e.g., using an accelerometer or a pedometer), we can use a more informative kinematic potential [118].

Now we can extend the BP method described in Section 3.2 for mobile networks. Denote the belief of node n at time t, in last iteration of static BP (see (3.7)), as $M^S_{n,t}(x_{n,t})$. To adapt the graphical model to mobile networks (according to Figure 4.1 and equation (3.7)), we can write the belief of node n in mobile networks as:

$$M^{FS}_{n,t}(x_{n,t}) = m_{t-1,t}(x_{n,t}) M^S_{n,t}(x_{n,t}) m_{t+1,t}(x_{n,t}) = M^F_{n,t}(x_{n,t}) m_{t+1,t}(x_{n,t}) \tag{4.3}$$

where $m_{t-1,t}(x_{n,t})$ represents the *filtering message* (i.e., the message from past to present), $m_{t+1,t}(x_{n,t})$ the *smoothing message* (i.e., the message from future to present), and $M^{FS}_{n,t}(x_{n,t})$ is the belief which includes the filtering and the smoothing messages. This belief can be available after N_t time frames (for N_t-leg smoothing). We will focus on 1-leg smoothing which can provide $M^{FS}_{n,t}(x_{n,t})$ at time $t+1$. By excluding $m_{t+1,t}(x_{n,t})$, we can also define the filtered belief $M^F_{n,t}(x_{n,t})$ which is available in real-time.

Using message update rule (see eq. (3.8)), we can define the filtering message (from $t-1$ to t), and the smoothing message (from $t+1$ to t). They are, respectively, given by:

$$m_{t-1,t}(x_{n,t}) \propto \int_{x_{n,t-1}} \psi_{t-1,t}(x_{n,t-1}, x_{n,t}) \frac{M^{FS}_{n,t-1}(x_{n,t-1})}{m_{t,t-1}(x_{n,t-1})} dx_{n,t-1} \tag{4.4}$$

$$m_{t+1,t}(x_{n,t}) \propto \int_{x_{n,t+1}} \psi_{t+1,t}(x_{n,t+1}, x_{n,t}) \frac{M^{FS}_{n,t+1}(x_{n,t+1})}{m_{t,t+1}(x_{n,t+1})} dx_{n,t+1} \tag{4.5}$$

Using (4.3), we can simplify previous equations:

$$m_{t-1,t}(x_{n,t}) \propto \int\limits_{x_{n,t-1}} \psi_{t-1,t}(x_{n,t-1}, x_{n,t})\, M^F_{n,t-1}(x_{n,t-1}) dx_{n,t-1} \tag{4.6}$$

$$m_{t+1,t}(x_{n,t}) \propto \int\limits_{x_{n,t+1}} \psi_{t+1,t}(x_{n,t+1}, x_{n,t})\, M^S_{n,t+1}(x_{n,t+1}) m_{t+2,t+1}(x_{n,t+1}) dx_{n,t+1} \tag{4.7}$$

Moreover, since we prefer to use 1-leg smoothing, we discard further information from the future (i.e., we set $m_{t+2,t+1}(x_{n,t+1}) = 1$):

$$m_{t+1,t}(x_{n,t}) \propto \int\limits_{x_{n,t+1}} \psi_{t+1,t}(x_{n,t+1}, x_{n,t})\, M^S_{n,t+1}(x_{n,t+1}) dx_{n,t+1} \tag{4.8}$$

Regarding nonparametric approximation, we can reuse the weighted particles from the static scenario $\{W^{S,j}_{n,t}, X^{S,j}_{n,t}\}$ (see Section 3.3), and use the filtering and the smoothing message to reweight them:

$$W^{FS,j}_{n,t} = m_{t-1,t}(X^{S,j}_{n,t}) \cdot W^{S,j}_{n,t} \cdot m_{t+1,t}(X^{S,j}_{n,t}) = W^{F,j}_{n,t} \cdot m_{t+1,t}(X^{S,j}_{n,t}) \tag{4.9}$$

Messages, (4.6) and (4.8), can be computed via Monte Carlo integration, i.e.:

$$m_{t-1,t}(x_{n,t}) \propto \sum_j \psi_{t-1,t}(X^{S,j}_{n,t-1}, x_{n,t})\, W^{F,j}_{n,t-1} \tag{4.10}$$

$$m_{t+1,t}(x_{n,t}) \propto \sum_j \psi_{t+1,t}(X^{S,j}_{n,t+1}, x_{n,t})\, W^{S,j}_{n,t+1} \tag{4.11}$$

As we can see, this method is very flexible, since there are three different beliefs available: $M^S_{n,t}$, $M^F_{n,t}$, and $M^{FS}_{n,t}$. The belief $M^F_{n,t}$ should be used in real-time applications, while the belief $M^{FS}_{n,t}$ should be used in all applications in which we can afford waiting one more sampling interval (T_S). On the other hand, the belief $M^S_{n,t}$ should not be used for tracking, but it is useful for testing the target dynamic (for example, it can be used to learn V_{max}, if not known in advance).

Finally, it is important to mention that the proposed extension of NBP is not exact in networks with loops. As explained in Chapter 3, it can cause overconfident beliefs of the position estimates. This problem, which is inherited from the static networks, can be solved using all proposed solutions in Chapter 3. However, in this chapter, we consider the networks with a negligible number of loops, so NBP-based methods will be good enough for the all analyses.

4.2.2 A novel communication protocol

Our second goal is to decrease the communication cost of NBP by: i) broadcasting the beliefs instead of the messages, ii) approximating the packages without a significant effect on the localization performance, and iii) avoiding the transmission of uninformative data. The proposed solutions are applicable for the mobile as well as for the static networks (Chapter 3).

Broadcasting beliefs

Naturally, one can assume that the messages should be transmitted between each pairs of the neighboring nodes. However, this produce a huge communication overhead since each node would need to send one message to each of the neighbors. Obviously, the main problem of this approach is that it does not take advantage of the broadcast in WSN (i.e., the transmitted message is needed at just one neighbor, not all). If we recall equation (see (3.9) in Chapter 3), we can see that the particles of the messages are constructed using particles from the: i) current belief of the node which transmits the message (source node), ii) measured distance, and iii) random angle. Since the samples of the distance can be measured by each node (prior to localization) and stored into memory, they should not be transmitted. The samples of the angles are drawn from the uniform distribution (3.9), so they can be computed at the destination. Thus, only particles of the beliefs, which are not available at the destination node, should be transmitted. One problem could be reweighting (3.10), using outgoing message from previous iteration, since each node has only incoming messages. It can be solved by computing the messages twice: once at the source node, and once at the destination node. This protocol is summarized in Alg 8.

The main benefit of this approach is that each node has to broadcast only one package[1] instead of n_d packages in case of message transmission (where n_d is node degree). This is paid by slight increase in computation since the messages must be computed twice. However, it is already well-known [76] that the communication is much more energy-consuming than computation.

Package Approximation

For the described protocol, we would need to transmit N_p particles (i.e., N_p weights, and $2N_p$ coordinates). However, we can avoid this using the following approximations:

[1]To avoid confusion, we use term "packages" for the scalar data that will be transmitted, in contrast to term "messages" which refers to NBP messages, which are never transmitted. For this analysis, the package contains only one scalar value.

Algorithm 8 Communication protocol (without approximation and censoring)

```
for all nodes do
    Obtain sufficient number of distance samples (from each neighbor)
    Initialize all belief and messages (see Section 3.2 in Chapter 3)
    for all iterations do
        Compute particles from outgoing messages and reweight them
        Transmit particles of the current belief
        Compute particles from incoming messages and reweight them
    end for
end for
```

- We resample with replacement before transmission in order to avoid transmission of the weights.
- The bandwidth (3.11) can be computed using the unweighted set of particles [13, 46], so it should not be transmitted.
- We approximate unweighted particles with Gaussian mixture, and transmit only their parameters. Upon receiving, we re-draw the set of particles from this mixture.

Since first two approximations are already part of the standard NBP (see Section 3.3.2), they do not affect accuracy. Regarding the last approximation, we expect that (given sufficient mixture components) it will not affect significantly the localization performance. Since the main problem of cooperative localization is the presence of multi-modal beliefs (caused by non-rigid graphs and/or multi-modal measurement noise), we expect that Gaussian mixture of very few components is appropriate choice. We can cluster unweighted set of particles using k-means algorithm [59], or expectation-maximization (see [9], chapter 9). The latter one can provide slightly better results, but with higher complexity. Thus, we recommend the use of k-means, especially for mobile networks.

Package censoring

We can additionally decrease communication using package censoring, i.e., by avoiding the transmission of the packages which provide little information. To that end, we do the following:

- In the first iteration, we only transmit the bounds of the bounded box (i.e., 4 scalar values, which define the rectangle) (see Section 3.3.4). Then, the particles can be drawn at the destination node.

- We do not transmit beliefs in the last iterations since they will never be used to update messages.
- Packages from anchor nodes are never transmitted (except their coordinates, if not known in advance).
- We do not transmit beliefs at iteration i which are similar to the beliefs in iteration $i-1$. The similarity can be measured using the KL divergence.
- We do not transmit parameters of mixture components which have very small weights (less than some predefined threshold).

We expect that these steps will, without any effect on accuracy, significantly decrease the communication cost. Note that we can also avoid receiving packages, as done in [26]. This technique should be applied if receiving the data is energy-consuming, and also in the case of single-cast communication.

4.2.3 Improving sampling techniques

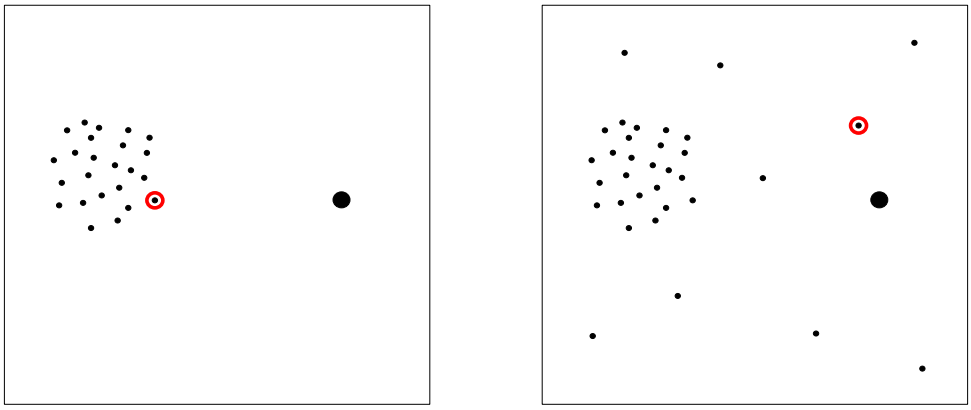
In this section, our goal is to improve the sampling techniques used in standard NBP. We propose three techniques: i) MIS with reference particles (MIS-RP), ii) PMC, and iii) the method based on an auxiliary variable.

MIS-RP

The MIS technique defined by (3.16) usually provides a very good set of particles, and outperforms a number of techniques as shown in [47, 48]. However, this might not be the case in some rare events, e.g., in the presence of the huge outliers (e.g., if obstacles are moving around).

According to the results in mobile robot localization [36], it is always useful to add a small number of uniformly distributed particles. These particles are essential for re-localization in the rare event that the sensor loses track of its position. We call these additional particles, *reference particles* (RP). In our case, this will especially happen if the messages from the neighboring target nodes provide wrong particles, but either anchor nodes or the kinematic message provide good weights. The problem is illustrated in Figure 4.2. Without RP (Figure 4.2a), MIS provides the set of particles in which the best candidate is very far from the true position (e.g., due to the outliers). With RP (Figure 4.2b), additional particles have been added uniformly in the whole area, so the best candidate is closer[2] to the true position.

[2]To simplify the example, we assumed the MAP estimate, but the same conclusion is valid for the MMSE estimate.



(a) (b)

Figure 4.2: Possible positions of target nodes in case of (a) MIS, and (b) MIS-RP. The true position of the target node is marked with black circle, and the best particle candidates are encircled.

Therefore, the importance density for MIS (at iteration $i+1$) can be written in the form:

$$q^{i+1}(x_u) = \sum\nolimits_{v \in G_u^0} m_{vu}^{i+1}(x_u) + \delta_{RP} \cdot Unif(x_u) \tag{4.12}$$

where $Unif(x_u) \propto 1$ within deployment area, $Unif(x_u) = 0$ otherwise, and δ_{RP} is the weight of the uniform distribution (in other words, the percentage of reference particles). δ_{RP} should be small (e.g., 10-20%) in order to keep computational cost reasonable. Note also that this importance density is legitimate since it is non-zero at places where the distribution that is being approximated is non-zero. Thus, in case of regular situations (when messages provide good particles), these additional particles will not cause any problem (i.e., after reweighting, their weights will be close to zero).

PMC

PMC [17, 19] is an iterative importance sampling technique where the importance density changes with every iteration in order to produce particles that better represent the target distribution. The standard importance sampling technique is a special case of PMC by running just one iteration. The general form of PMC is illustrated in Alg. 9.

In order to use PMC for cooperative localization, we need to choose the importance density that we want to improve. We choose the density used for MIS (given by (3.16)) or MIS-RP (given by (4.12)) as prior. Distribution $p(X_u^{j,m})$ used for reweighting in Alg. 9 is given by the numerator of (3.15). For the KDE, we again use a spherical Gaussian Kernel with bandwidth h. Finally, in each iteration of Alg. 9,

Algorithm 9 Population Monte Carlo (PMC) (for node u)

1: Choose initial importance function: $q_u^1(x_u)$
2: **for all** iterations $m = 1 : N_m$ **do**
3: Draw particles: $X_u^{j,m} \sim q_u^m(x) \quad (j = 1...N_p)$
4: Compute weights: $W_u^{j,m} = \frac{p(X_u^{j,m})}{q_u^m(X_u^{j,m})}$
5: Normalize weights: $W_u^{j,m} = \frac{W_u^{j,m}}{\sum_j W_u^{(m)}}$
6: Resample with replacement
7: Update importance density: $q_u^{m+1}(x_u) = \sum_j \mathcal{K}_h(x_u - X_u^{j,m})$
8: **end for**

we can draw a new set of the particles from this kernel as follows:

$$X_u^{j,m} = X_u^{j,m-1} + \epsilon_u^j \cdot [\cos(\theta_u^j) \;\; \sin(\theta_u^j)] \tag{4.13}$$

where $\epsilon_u^j \sim N(\epsilon; 0, h)$ and $\theta_u^j \sim Unif[0, 2\pi)$. Note that we used simplified notation by removing the NBP iteration (do not mix NBP iterations with PMC iterations). We refer to this version of NBP, as PMC-NBP.

Finally, we propose an optional approximation of the PMC-NBP method. The main computational problem of NBP and variations is the computation of KDE, which requires $O(N_p^2)$ operations. This is especially the problem in PMC-NBP, since it has nested iterations (i.e., within one NBP iteration, there are N_{PMC} iterations). Therefore, instead of using full information (the product of the messages from all the neighbors), we use only information from the anchors. In order to keep the NBP algorithm regular, we just need to keep full information in the first iteration of the PMC (which corresponds to the standard importance sampling). Since we do not use information from the target nodes, this method represents a *non-cooperative* PMC. Note that this approach will not only improve the beliefs of the anchors' neighbors. Since NBP is still a cooperative method, in NBP's very next iteration, the improved estimate of the anchors' neighbors will be flooded further into the network.

Auxiliary variable

Standard importance sampling used in NBP does not take into account most of the available information in the graph. This often causes high variance of the weights, i.e. there will be a lot of particles in the regions of low probability, and very few (or even just one) in the regions of high probability. One solution to this problem is to use the optimal importance density, which includes all the available information, but this is not feasible in most cases [2]. A second solution is PMC from the previous section, which iteratively improves the importance density. The auxiliary

particle filtering (APF) [35,81] is an alternative solution which tries to predict (using auxiliary variable) which particles will be in regions of high probability.

However, NBP is a generalization of particle filtering for any graphical model, so we need to adapt the standard APF method. One framework has been already proposed in [16], in which the authors propose to use the index of the messages as an auxiliary variable in order to predict which message provides better information. This method is not very suitable (especially, for localization) due to the high dimensional auxiliary variable. In contrast to this approach, we will choose a 1D auxiliary variable, which will provide the largest amount of information.

Let us recall equation (3.9), written in a more general form:

$$x_{ru}^{j} = X_{r}^{j_X} + (d_{ru} + v^{j_d})[\cos(\theta^{j_\theta}) \;\; \sin(\theta^{j_\theta})] \tag{4.14}$$

in which we again removed the NBP iteration index and, in contrast to (3.9), we use a different index ($j, j_X, j_d, j_\theta \sim 1...N_p$) for each random variable. As we can see, to update the particles of the messages (x_{ru}^{j}), we need to use three random variables: particles of the current position ($X_r^{j_X} \sim M_r(x_r)$), distance samples ($d_{ru} + v^{j_d} \sim p_v$), and angle samples ($\theta^{j_\theta} \sim Unif[0, 2\pi)$). The auxiliary variable could be an index of any of these three random variables. Obviously, the uniformly distributed θ^{j_θ} includes the smallest amount of information (the entropy is maximal) than any other random variable, so we choose j_θ as the auxiliary variable. Then, we can set the other indices to the same value ($j_X = j_d = j$). In other words, instead of drawing samples uniformly in any direction (which will create a lot of particles with small weights), we will draw them in the most likely direction according to the distribution of the auxiliary variable. To achieve this, we first, for each index j_θ, find some likely value associated with the message, e.g., expected value:

$$\mu_{ru}^{j_\theta} = \mu_{X_r} + \mu_{d_{ru}}[\cos(\theta^{j_\theta}) \;\; \sin(\theta^{j_\theta})] \tag{4.15}$$

where we averaged the left-hand side of (4.14) over j. The computed set of mean values is further used to compute *first-stage* (1st) weights:

$$w_{ru}^{j_\theta,1st} \propto p(Y|\mu_{ru}^{j_\theta}) \tag{4.16}$$

which represent the likelihood function given all information (Y) that we want to include. Recall that these weights are for the message from node r to node u, which represents some information about position of node u. Thus, we can include the product of all the messages coming to node u, but as for PMC, we again restrict to information from the anchors. For this approach, it is even more critical because the

number of messages (equal to twice the number of the edges in the graph) is typically significantly larger than the number of target nodes. Therefore, the likelihood of the information that we want to include is given by:

$$p(Y|\mu_{ru}^{j_\theta}) = \prod_{a\in G_a} p(d_{au}|x_a^*, \mu_{ru}^{j_\theta}) \propto \prod_{a\in G_a} \psi_{au}(x_a^*, \mu_{ru}^{j_\theta}) \tag{4.17}$$

where G_a is the set of all the anchor neighbors of node u. In case of no anchors in the neighborhood, we simply do not apply this approach. First-stage weights provide us information on how likely is the index of the angle j_θ. Therefore, given the multinomial distribution defined by first-stage weights, we can draw set of N_p indices $ind(j_\theta)$ $(j_\theta = 1...N_p)$. Then, we can compute particles from the message:

$$x_{ru}^{j_\theta} = X_r^{j_\theta} + (d_{ru} + v^{j_\theta})[\sin(\theta^{ind(j_\theta)}) \;\; \cos(\theta^{ind(j_\theta)})] \tag{4.18}$$

Finally, the whole procedure is still not regular due to the double-counting of the information from anchors (which is regularly used in (3.15)). Thus, the regular weights, given by (3.10), should be divided by the weights of the importance density given by the first-stage weights (4.16). The final weights for the particles from the message, also called the *second-stage* weights [81] are given by:

$$w_{ru}^{j_\theta} = \frac{W_r^{j_\theta}}{m_{ur}(X_r^{j_\theta})} \cdot \frac{1}{p(Y|\mu_{ru}^{ind(j_\theta)})} \tag{4.19}$$

Given these weights, we can proceed with the standard NBP. We refer to this version as *auxiliary* NBP (ANBP). It is worth noting that the standard NBP is a special case of ANBP, if no additional information (Y) has been used, i.e., when $p(Y|\mu_{ru}^{j_\theta}) \propto 1$, and $p(\theta^{ind(j_\theta)}) = p(\theta^{j_\theta}) \propto Unif[0, 2\pi)$.

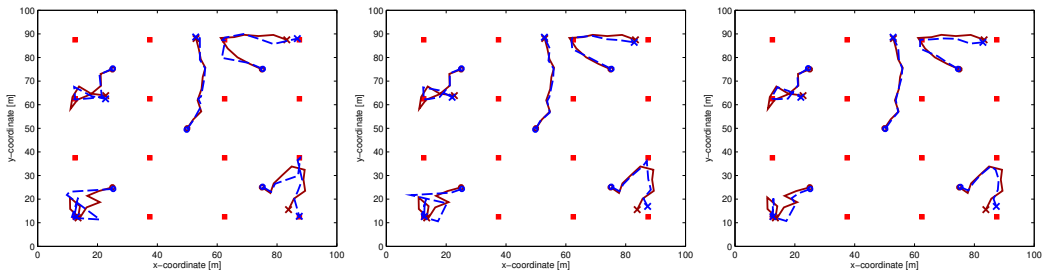
4.2.4 Simulation results

We conducted several simulations to analyse the performance of the NBP method in mobile networks, the effect of package approximation and censoring, and the effect of the improved sampling techniques.

Analysis of mobile positioning based on NBP

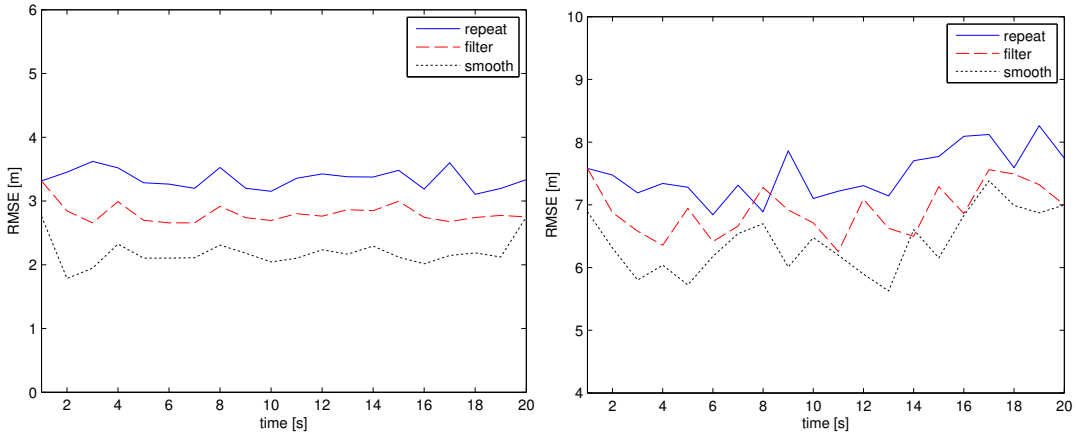
In first set of tests, we assume that there are $N_a = 16$ anchor nodes and $N_t = 5$ target nodes, deployed in a 100m x 100m area. Anchor nodes are deployed in grid, or semi-random[3] topology. Target nodes are moving according to the Gaussian-

[3]The area is divided into N_a square-shaped cells, and each anchor node is deployed randomly within one of them (i.e., one anchor per cell).



(a) (b) (c)

Figure 4.3: Tracking 5 nodes using (a) repeating of the static NBP method, (b) filtered NBP estimate, and (c) smoothed (1-leg) NBP estimate. Anchor nodes (in grid topology) are marked with squares, true track with lines, and estimated track with dashed lines (starting points of the tracks are marked with circles, and destination points with 'X').



(a) (b)

Figure 4.4: Comparison of the RMSE for: (a) grid, and (b) semi-random topologies of the anchor nodes.

Markov mobility model [18], which uses one tuning parameter to vary the degree of randomness of the movement, and can easily ensure that the target is always within the deployment area. The parameters are set to the following values: number of particles $N_p = 400$, communication radius $R = 20$m, standard deviation of the zero-mean Gaussian noise for the measured distance $\sigma_d = 1$m, sampling interval $T_S = 1$s, tracking period $T_P = 20$s, maximum speed $V_{max} = 5$m/s, number of iterations $N_{iter} = 3$, and number of Monte Carlo runs $N_{mc} = 100$.

In Figures 4.3a-4.3c, we show an example of estimated tracks for three different NBP estimates. As we can see, all the estimates are similar in the case of a sufficient number of neighbors, but the smoothed estimate is the best for the tracks close to the edges. We also compare RMSE of all the three methods for three different deployments (Figure 4.4). As expected, the smoothed estimate consistently performs

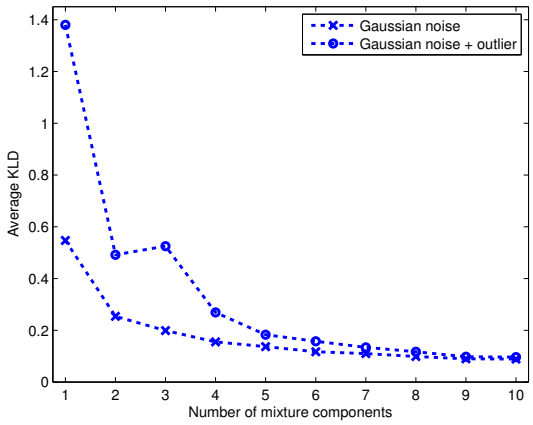


Figure 4.5: KLD between approximated belief and particle-based belief as function of number of mixture components.

better than the filtered estimate, which performs better than naive repeating of the static NBP localization method. Note that the smoothed (1-leg) estimate is available 1 second after filtered estimate, but this delay should not be a problem for most applications. On the other hand, we can see that the deployment of the anchor nodes significantly affects accuracy. Thus, it is advisable to use the grid deployment. However, if it is not possible, this problem can be solved (with an additional cost) either by increasing the number of anchors or by increasing the communication radius.

Analysis of package approximation and censoring

In contrast to previous tests, we change the number of nodes ($N_a = 5$, $N_t = 10$), tracking period $T_P = 5$s, and the communication radius ($R = 40$m). The anchors are deterministically placed (4 near the edges, and one in the center). To make the analysis more general, in some simulations we add an outlier component (25% of the true distance) to the zero-mean Gaussian noise. More precisely, the noise is a two-component Gaussian mixture with the same weights ($w_{d,1} = w_{d,2} = 0.5$) and same standard deviations ($\sigma_{d,1} = \sigma_{d,2} = 1$m), but different means ($\mu_{d_{ru},1} = 0$, $\mu_{d_{ru},2} = 0.25 d_{ru}$). This noise is applicable to the scenario in which there is an obstacle (between two sensors) for 50% of the time. Moreover, taking into account the conclusion from the previous section, we use the smoothed estimate of the NBP.

We start by analysing the KLD between the approximated belief and the particle-based belief w.r.t. the number of mixture components N_m. According to Figure 4.5, we can see that KLD is decreasing as we increase N_m, as expected. In the case of Gaussian noise, we just need 3 or 4 mixture components[4], but if we add an outlier,

[4]Note that Gaussian noise does not necessarily lead to Gaussian posteriors due to the nonlinearity

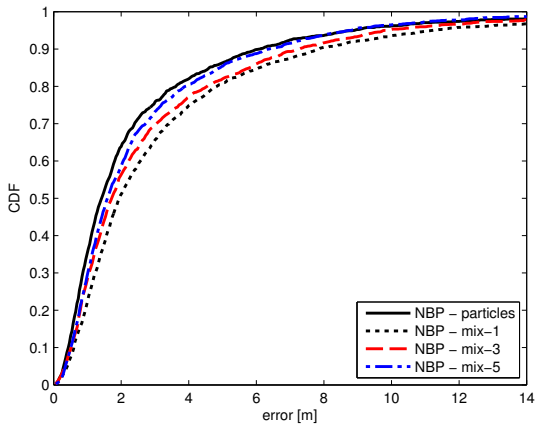


Figure 4.6: CDF of the position error for different approximations.

we will need a few more mixture components. To see how this approximation affects the error, we analyse the CDF for different approximations of the beliefs (particle-based, and beliefs represented with $N_m = 1$, $N_m = 3$, $N_m = 5$ mixture components). We consider the case with outlier since this is more critical case. As we can see in Figure 4.6, 5-mixture approximation achieve almost the same accuracy as the particle-based approximation. However, the number of mixture components should be a tuning parameter, which will allow the user to make the trade-off between accuracy and cost. Note also that package censoring proposed in Section 4.2.2 does not affect accuracy at all, assuming that KLD threshold (used for measuring the similarity between beliefs) is sufficiently small (less than 0.2, in our case).

Finally, we analyse the communication cost per node within one time frame. According to Table 4.1, we can conclude the following:

- New protocol (Alg. 8) decrease the cost n_d times. In our case (where $n_d \approx 2.56$, excluding anchors), instead of transmission of 6144 packages (scalar values), we need to transmit 2400 packages.

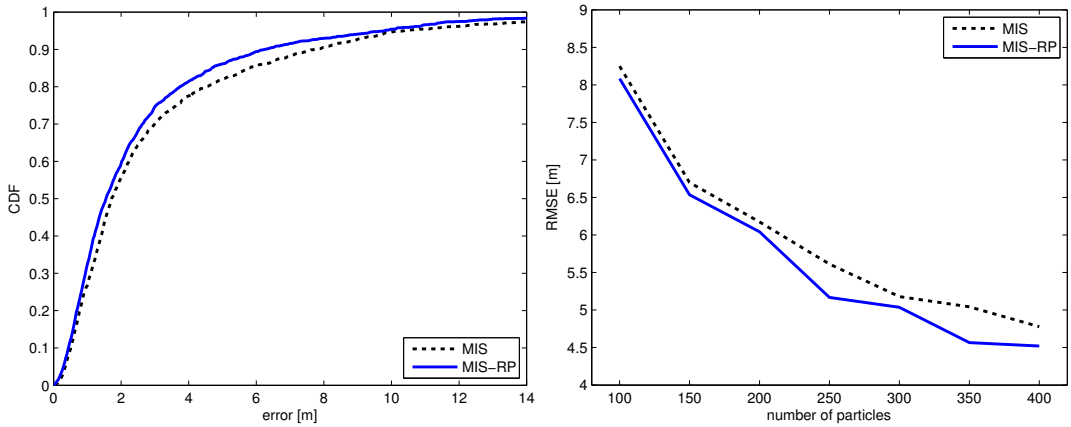
- Mixture approximation significantly decreases communication cost (97%). In our case (with $N_m = 5$), we just need to transmit the parameteres: mean values ($2N_mN_{iter} = 30$ packages), the variances ($2N_mN_{iter} = 30$ packages, assuming diagonal covariance matrix), and the weights ($(N_m - 1)N_{iter} = 12$ packages) .

- Message censoring also decreases the communication (37% in our case), especially because of the savings in the first and the last iteration.

(especially problem in non-rigid graphs).

Table 4.1: Number of transmitted packages (N_{pack}) for different protocols and approximations.

Package	N_{pack}
message	6144
belief (Alg. 8)	2400
belief (5-mix approx.)	72
belief (5-mix approx. and censoring)	45



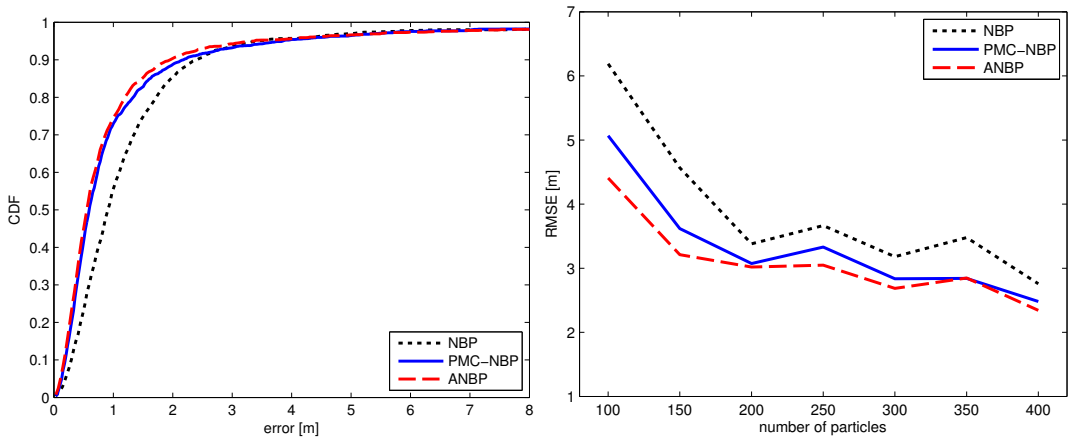
(a) (b)

Figure 4.7: Comparison between MIS and MIS-RP: (a) CDF of the position error (400 particles used), and (b) RMSE as function of number of particles.

For the further analysis, we will assume that each belief is approximated with a Gaussian mixture of 5 components, since it practically has no effect on accuracy.

Analysis of sampling techniques

We start with the comparison between MIS and MIS-RP techniques. We consider the same scenario as in previous section (with outliers). For MIS-RP, we added 20% reference particles, which are uniformly distributed within the deployment area. According to Figure 4.7a, we can see that the MIS-RP consistently outperforms the MIS technique. For example, 90th percentile is about 2m less in case of MIS-RP. We also compared RMSE w.r.t. the number of particles. As we can see in Figure 4.7b, with MIS-RP technique, we decrease the error up to 0.5m. More importantly, by applying MIS-RP we can achieve the same error (e.g., 5m) using 15-25% fewer particles. This practically compensates previously added reference particles. It is also worth to mention that MIS-RP performs similarly to MIS if there are no outliers. In any case, it is strongly recommended to use MIS-RP in order to increase the robustness of the localization method.



(a) (b)

Figure 4.8: Comparison between NBP, PMC-NBP and ANBP methods: (a) CDF of the position error (400 particles used), and (b) RMSE as function of number of particles.

We now provide the comparison between NBP, PMC-NBP, and ANBP method. We again consider the same scenario as in previous section, but without outliers. For PMC-NBP, we found that it is sufficient to use 5 PMC iterations. In Figure 4.8a, we compare the CDF for all three techniques. As expected, PMC-NBP, and ANBP provides more accurate estimate (about 0.5m, in our case). Moreover, both methods (PMC-NBP and ANBP) provide nearly the same estimate. This is expected since we used the same information to improve the importance density (information from the anchors). In Figure 4.8b, we provided a comparison of the RMSE w.r.t. the number of particles. We can see that the benefit of PMC-NBP and ANBP can be even up to 1m if we use less particles. Moreover, ANBP can outperform PMC-NBP for a small number of particles. We also note that if we need to achieve predefined accuracy (e.g., 3m), we can use significantly fewer particles (15-30%, in our case). That means that PMC-NBP /ANBP are more efficient than standard NBP.

Finally, the main question is which method should be applied (PMC-NBP or ANBP) since both of them provide similar performance. The communication cost of both PMC-NBP and ANBP is the same as the NBP cost, since all modifications can be done locally. However, taking into account that ANBP tries to improve particles from the messages, and PMC-NBP particles from the beliefs, the latter one is less complex (assuming a small number of PMC iterations). Therefore, PMC-NBP should be applied for low-cost applications. Otherwise, ANBP should be used, since it is slightly more accurate than PMC-NBP.

4.3 Distributed target tracking

Distributed tracking[5] in WSN is important tasks for many applications in which central unit is not available. For example, in emergency situations, such as fire or nuclear disaster, WSN can be deployed to detect these phenomena. Once the phenomena is detected (e.g., increased temperature, or radioactivity), the sensors start to sense their neighborhood and cooperatively track people and assets. As sensors are low-cost devices that may not survive during deployment, it is important to achieve tracking in a manner that is fully asynchronous and robust to sensors failures, and in such a way that every sensor has the same belief of the target location. Moreover, due to the nonlinear relationships and possible non-Gaussian uncertainties, a particle filtering (PF) should be applied [2], instead of traditional methods based on Kalman filtering (KF) [116].

Most of the methods for PF-based distributed target tracking in WSN are based on the construction and maintenance of the communication path. For example, in [22], low-power sensors pass the parameters of likelihood function to the high-power sensors, which are responsible to manage the low-power nodes. In [103], a set of uncorrelated sensor cliques is constructed, in which slave nodes have to transmit Gaussian mixture parameters to the master node of the clique. Master node performs the tracking, and forward estimates to another clique. In [62], a Markov-chain distributed PF is proposed, which does not route the information through the graph during tracking. However, it requires that each node knows total number of communication links and the number of communication links between each pair of nodes, which can be obtained only by aggregating the data before tracking. These, *routing-based*[6] algorithms lack robustness to failures and are also not suitable for asynchronous networks. To address these problems, several authors have considered using average consensus algorithms. In [41], the global posterior distribution is approximated with a Gaussian mixture, and consensus is applied over the local parameters to compute the global parameters. Similarly, [42, 44] uses a Gaussian approximation instead of Gaussian mixture. Randomized gossip consensus was used in [74,106] for distributed target tracking. Finally, as a benchmark, we also mention the non-centralized PF (NCPF) [29], in which each node broadcasts measurements until all the nodes have complete set of measurements. Then, each node (acting like a fusion center) performs the tracking. Although this method is not scalable, it can be still competitive in some scenarios. These state-of-the-art methods suffer

[5]Note that distributed tracking is also cooperative. In order to be consistent with literature, we will refer to it as distributed tracking.

[6]Authors sometimes use the term "message passing" [41, 42, 62] for this type of methods. This can be confusing with the standard message passing method, belief propagation, which does not belong to this category.

from at least one of the following problems: i) they do not use the fastest consensus methods, and ii) they cannot handle all parametric and nonparametric likelihood functions.

In following sections, we propose and evaluate a general framework for target tracking using distributed particle filtering (DPF) based on three asynchronous belief consensus (BC) algorithms: standard belief consensus (SBC), broadcast gossip (BG), and belief propagation (BP). While parametric variants of DPF-SBC have been already used, DPF-BG and DPF-BP are, to the best of our knowledge, novel for distributed target tracking. We also determine when it is beneficial to use proposed DPF methods over NCPF, and provide extensive simulation results. Our main result is that DPF-BG and DPF-SBC provide the best performance in terms of RMSE, and that DPF-BP provides the best performance in terms of disagreement in the network.

4.3.1 Overview of centralized target tracking

We consider the problem of tracking a target in WSN. We assume that there is a number of static sensor nodes with known positions and one moving target (e.g., a person or vehicle) in some surveillance area. The target may be passive, but the sensors are assumed to periodically make observations that depend on the relative position of the target and the sensing node. The goal of the WSN is to track the position and velocity of the target. In this section, we describe a centralized approach to solve this problem, in which all the measurements are collected by a sensor that acts as fusion center. Although we focus on single-target tracking, the algorithm can be applied for multi-target tracking if the targets are labeled (e.g., using RFID; see Chapter 5). Otherwise, different algorithms should be applied [39, 58, 117].

System model

The scenario under consideration is illustrated in Fig. 4.9. There are N_s sensors with known two-dimensional (2D) positions, l_n $(n = 1, 2, \ldots, N)$ and one target with an unknown state x_t at time t. The state of the target is defined as $x_t = [x_{1,t}\ x_{2,t}\ \dot{x}_{1,t}\ \dot{x}_{2,t}]^T$, where $x_{1,t}$ and $x_{2,t}$ represent 2D position of the target, and $\dot{x}_{1,t}$ and $\dot{x}_{2,t}$ the 2D velocity of the target. The goal of the WSN is to estimate x_t at each (discrete) time t. We use the following state-space model:

$$x_{t+1} = Ax_t + Bu_t \tag{4.20}$$

$$y_{n,t} = g_n(x_t) + v_{n,t}, \tag{4.21}$$

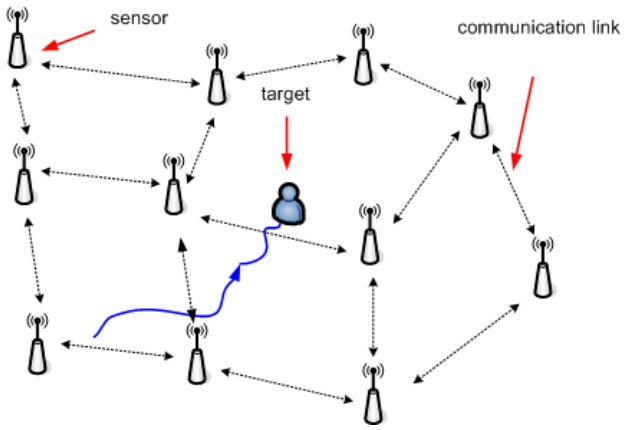


Figure 4.9: Illustration of target tracking in a WSN.

where $u_t = [u_{1,t}\, u_{2,t}]^T$ is the process noise due to the variation of the speed, $y_{n,t}$ is local observation of sensor n at time t, and $v_{n,t}$ is its observation noise. The process noise u_t can be non-Gaussian, but since it is usually hard to measure [94, 116], we assume a Gaussian approximation with sufficiently large variance (e.g., upper bound of real uncertainty), which is common choice. The matrices A and B are given by

$$A = \begin{bmatrix} \boldsymbol{I}_2 & T_S \boldsymbol{I}_2 \\ \mathbf{0}_2 & \boldsymbol{I}_2 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{T_S^2}{2} \boldsymbol{I}_2 \\ T_S \boldsymbol{I}_2 \end{bmatrix}, \tag{4.22}$$

where T_S is the sampling interval, and $\boldsymbol{I}_2$ and $\mathbf{0}_2$ represent the identity and zero 2 x 2 matrices, respectively. We denote by G_t the set of the nodes that have a measurement available at time t. For the sake of concreteness, we assume that the measurements are distance measurements to the target, i.e., for $n \in G_t$,

$$g_n(x_t) = \left\| l_n - [x_{1,t}\ x_{2,t}]^T \right\|. \tag{4.23}$$

The measurement noise $v_{n,t}$ is distributed according to $p_v(\cdot)$, which is not necessarily Gaussian, and typically depends on measurement technique (e.g., acoustic [1], RSS [71], RF tomography [20]) and the environment.

For simplicity, we assume ideal probability of detection for both sensing and communication range, but more complex models can be easily incorporated [46]. That means that a sensor can detect the target if the distance between them is less than predefined value r, and that two sensors can communicate with each other if the distance between them is less than R. Taking into account that radio of a node is usually much more powerful than its sensing devices [38, 51], we assume $R \geq r$.

Algorithm 10 CPF (at time t)

1: **for all** particles $m = 1 : N_m$ **do**
2: Draw particle: $x_t^{(m)} \sim p(x_t|x_{t-1}^{(m)})$
3: Compute weight: $w_t^{(m)} = w_{t-1}^{(m)} \cdot p(y_t|x_t^{(m)})$
4: **end for**
5: Normalize: $w_t^{(m)} = w_t^{(m)} / \sum_m w_t^{(m)}$ (for $m = 1 : N_m$)
6: Compute estimates: $\hat{x}_t = \sum_m w_t^{(m)} x_t^{(m)}$
7: Resample with replacement from $\{w_{n,t}^{(m)}, x_{n,t}^{(m)}\}_{m=1}^{N_m}$

Particle filtering

We apply the Bayesian approach for this tracking problem and recursively determine the posterior distribution $p(x_t|y_{1:t})$ given the prior $p(x_{t-1}|y_{1:t-1})$, dynamic model $p(x_t|x_{t-1})$ defined by (4.20), and the likelihood function $p(y_t|x_t)$ defined by (4.21). We assume that $p(x_0|y_0) = p(x_0)$ is initially available. The posterior can be found using the prediction and filtering equations [2]:

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})dx_{t-1} \tag{4.24}$$

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}). \tag{4.25}$$

Assuming independence among each measurements at time t, the global likelihood function $p(y_t|x_t)$ can be written as the product of the local likelihoods:

$$p(y_t|x_t) \propto \prod_{n \in G_t} p(y_{n,t}|x_t). \tag{4.26}$$

For notational convenience we will still write $p(y_{n,t}|x_t)$ for $n \notin G_t$, with the tacit assumption that this function is identically equal to 1.

Since the measurement noise is generally not Gaussian, and the measurement is not a linear function of the state, a traditional KF [2,116] approach can not be used. Instead, we apply the PF [2], in which the posterior distribution is represented by a set of samples (particles) with associated weights. A well-known solution is the *sample-importance-resampling* (SIR) method, in which the particles are drawn from $p(x_t|x_{t-1})$, then weighted by the likelihood function, $p(y_t|x_t)$, and finally resampled in order to avoid degeneracy problems (i.e., the situation in which all but one particle have negligible weights). More advanced versions of PF also exist [57, 81, 107], but we focus on SIR since the distributed implementation of most PF-based methods is similar. We will refer to PF with SIR as centralized PF (CPF). The CPF method is summarized in Alg. 10.

This algorithm is run on one of the nodes in the WSN, which serves as fusion center. The main drawbacks of the CPF are: i) large energy consumption on the nodes which are in proximity of the fusion center, ii) high communication cost in large-scale networks; iii) the posterior distribution cannot be accessed from any node in the network; and iv) fusion center has to know the locations, observations, and observation models of all the nodes. In the following section we will focus on distributed implementations of PF method, which alleviate these problems.

4.3.2 Distributed particle filtering

Our goal is to track the target in a distributed, asynchronous way, such that all the nodes have a common view of the state of the target. We use distributed implementation of the PF (DPF), in which we want to avoid exchanging measurements and to have a common set of samples and weights at every time step. If we can guarantee that the samples at time $t-1$ are common, and the weights at time t are common, then common samples at time t can be achieved by providing all nodes with the same seed for random number generation, so as to ensure that their pseudo-random generators are in the same state at all times. Ensuring common weights for all nodes can be achieved by means of a BC algorithm. BC formally aims to compute, in a distributed fashion the product of a number of functions over the same variable

$$\mathrm{BC}(f_1(x), f_2(x), \ldots, f_{N_s}(x)) = \prod_{n=1}^{N_s} f_n(x). \tag{4.27}$$

However, most BC algorithms are not capable to achieve exact consensus in a finite number of iterations (except BP-consensus in tree-like graphs; see Section 4.3.3). As we require exact consensus on the weights, we additionally apply max-consensus[7] (MC) [72, 106],

$$\mathrm{MC}(f_1(x), f_2(x), \ldots, f_{N_s}(x)) = \max_n f_n(x), \tag{4.28}$$

which computes the exact maximum over all arguments using the same asynchronous protocol as average consensus in a finite number of iterations (equal to the diameter of the graph). This idea has been already used in [106] for gossip-based consensus. The final algorithm is shown in Alg. 11. Observe that, in contrast to CPF, the following drawbacks have been removed: i) energy consumption is balanced across the network; ii) reduced communication cost in certain scenarios (see later in the chapter); iii) every node has access to the posterior distribution; and iv) no knowledge required of the locations, observations, or observation models of any other node.

[7] Min-consensus can be also applied.

Algorithm 11 DPF (at node n, at time t)

1: **for all** particles $m = 1 : N_m$ **do**
2: Draw particle: $x_t^{(m)} \sim p(x_{n,t}|x_{t-1}^{(m)})$
3: Compute weight: $w_{n,t}^{(m)} = w_{t-1}^{(m)} \cdot \mathrm{BC}\left(p(y_{1,t}|x_t^{(m)}), ..., p(y_{N_s,t}|x_t^{(m)})\right)$
4: **end for**
5: Normalize: $w_{n,t}^{(m)} = w_{n,t}^{(m)} / \sum_m w_{n,t}^{(m)}$ (for $m = 1 : N_m$)
6: Compute estimates: $\hat{x}_{n,t} = \sum_m w_{n,t}^{(m)} x_{n,t}^{(m)}$
7: $\hat{w}_t^{(m)} = \mathrm{MC}\left(w_{1,t}^{(m)}, ..., w_{N_s,t}^{(m)}\right)$ (for $m = 1 : N_m$)
8: Normalize: $\hat{w}_t^{(m)} = \hat{w}_t^{(m)} / \sum_m \hat{w}_t^{(m)}$ (for $m = 1 : N_m$)
9: Resample with replacement from $\{\hat{w}_t^{(m)}, x_t^{(m)}\}_{m=1}^{N_m}$

In the next section, we will describe three distinct BC algorithms.

4.3.3 Belief consensus algorithms

Our goal is to approximate the product of the local likelihoods using BC algorithms. Motivated by their scalability, asynchronous behavior and robustness to failures [6, 24, 73], we consider three variants of BC: SBC [73], BC based on BG [6], and BC based on BP [24, 77].

SBC

SBC [73] is defined in following iterative form:

$$M_n^{(i)}(x_t) = M_n^{(i-1)}(x_t) \prod_{u \in G_n} \left(\frac{M_u^{(i-1)}(x_t)}{M_n^{(i-1)}(x_t)} \right)^{\epsilon}, \tag{4.29}$$

where G_n is the set of neighbors of node n, $M_n^{(i)}$ represents current estimate (at iteration i) of the global likelihood of the variable x_t (in our case, $x_t \in \{x_t^{(1)}, \ldots, x_t^{(N_m)}\}$), and ϵ depends on maximum node degree in the network ($0 < \epsilon < 1/\eta_{\max}$, where $\eta_{\max}$ is maximum node degree in the network). For convenience, we define the *update rate* ξ ($0 < \xi < 1$), so $\epsilon = \xi/\eta_{\max}$. Update rate $\xi \approx 1$ is expected to provide the fastest convergence [72]. Note that logarithm of (4.29) represents standard average consensus algorithm [72]. We initialize by

$$M_n^{(1)}(x_t) = p(y_{n,t}|x_t). \tag{4.30}$$

This consensus algorithm *guarantees* convergence (in all connected graphs) as number of iterations goes to infinity [73]. Thus, it asymptotically converges to the geometrical average of the local distributions:

$$\lim_{i\to\infty} M_n^{(i)}(x_t) = \left(\prod_{n\in G_t} p(y_{n,t}|x_t) \right)^{1/N_s}, \tag{4.31}$$

from which the desired quantity, $\prod_{n\in G_t} p(y_{n,t}|x_t)$, can easily be found, for any value of $x_t \in \{x_t^{(1)}, \ldots, x_t^{(N_m)}\}$.

If the maximum node degree ($\eta_{\max}$) and number of nodes (N_s) are not known a priori, we need to estimate them in distributed way. The estimation of maximum node degree can be done using max-consensus, while N_s can be determined [80] by setting the initial state of one node to 1, and all others to 0. By using average consensus [72], they can obtain the result $1/N_s$, which is the inverse of the number of nodes in the network. We refer to this method as DPF-SBC.

BC based on BG

Gossip-based algorithms [28] can also achieve consensus in asynchronous networks. In order to use the broadcast nature of WSN, we choose *broadcast gossip* (BG) [6]. It has been shown [6] that this method is significantly faster than other well-known gossip-based methods, such as randomized gossip [15], and geographic gossip [27], in which only one pair of the nodes update its state per iteration. In broadcast gossip, it is assumed that all the nodes has internal clock which ticks independently according to a rate of e.g., a Poisson process [6]. When the clock of the n-th node ticks, node n broadcasts its own state value. This state value is received by all neighbors within communication radius R. Then, these nodes will make weighted average of their current state value and the received state value. It has been shown [6] that BG converges, in expectation, to the real average value.

For the belief consensus, we need to achieve convergence to the geometrical average (4.31), so at the k-th clock tick of node n all the nodes make the following operation:

$$M_u^{(k)}(x_t) = \begin{cases} M_u^{(k-1)}(x_t)^{\gamma} M_n^{(k-1)}(x_t)^{1-\gamma}, & u \in G_n \\ M_u^{(k-1)}(x_t), & \text{otherwise} \end{cases} \tag{4.32}$$

where $0 < \gamma < 1$ is the *mixing parameter*. It has been shown [6] that optimal value of γ depends on the *algebraic connectivity* of the graph (which represent the second smallest eigenvalue of the Laplacian matrix [6, 72]). However, this parameter is not available in distributed scenario, so empirical study has been used [6] to find the

optimal value of γ. Therefore, we will model γ as function of average node degree $\bar{\eta}$ in the network, since $\bar{\eta}$ can be easily estimated in distributed way.

Initialization is exactly the same as for SBC. We also need to apply average consensus to estimate N_s and $\bar{\eta}$. We refer to this variant of DPF as DPF-BG.

However, we make one difference comparing with standard BG. In order to have the same communication cost per iteration, we assume that one SBC iteration corresponds to N_s BG iterations (i.e., $i = \lceil k/N_s \rceil$). This assumption is reasonable taking into account that, to avoid collisions, even SBC has to broadcast its data in sequential way.

BC based on BP

BP is well-known message passing algorithm on an undirected graphical model (see [24, 77] and Chapter 3). Consider the following function:

$$\prod_n p(y_{n,t}|x_{n,t}) \prod_{u \in G_n} \delta(x_{n,t} - x_{u,t}), \tag{4.33}$$

which is equal to $\prod_{n \in G_t} p(y_{n,t}|x_t)$, whenever all the dummy variables are the same. Comparing (4.33) with (3.6)[8], we can see that, if we set pairwise potential to delta Dirac impulse, running BP on the corresponding graph yields the marginals $M_n(x_{n,t}) = C \prod_n p(y_{n,t}|x_{n,t})$ for every n, where C is a normalization constant. Note that this normalization constant is irrelevant as weights in Alg. 11 will be normalized later anyway. The BP message passing equations are now as follows: the belief at iteration i (the current approximation of $C \prod_n p(y_{n,t}|x_{n,t})$) is, according to (3.7), given by:

$$M_n^{(i)}(x_{n,t}) \propto p(y_{n,t}|x_{n,t}) \prod_{u \in G_n} m_{un}^{(i)}(x_{n,t}), \tag{4.34}$$

while the message from node $u \in G_n$ to node n is, according to (3.8), given by:

$$m_{un}^{(i)}(x_{n,t}) \propto \int_{x_{u,t}} \delta(x_{n,t} - x_{u,t}) \frac{M_u^{(i-1)}(x_{u,t})}{m_{nu}^{(i-1)}(x_{u,t})} dx_{u,t} = \frac{M_u^{(i-1)}(x_{n,t})}{m_{nu}^{(i-1)}(x_{n,t})}. \tag{4.35}$$

Previous equation (4.35), can be written as:

$$m_{un}^{(i)}(x_t) \propto \frac{M_u^{(i-1)}(x_t)}{m_{nu}^{(i-1)}(x_t)}, \tag{4.36}$$

[8]Note that, in this chapter, t is time index, in contrast to Chapter 3.

where we removed index n since all the nodes have the same variable ($x_{n,t} = x_{u,t} = x_t$). The denominator of (4.36) is the message from node n to node u in the previous iteration, and can be expressed as

$$m_{nu}^{(i-1)}(x_t) \propto \frac{M_n^{(i-2)}(x_t)}{m_{un}^{(i-2)}(x_t)}. \tag{4.37}$$

Combining previous two equations, we get the recursive expression for the messages:

$$m_{un}^{(i)}(x_t) \propto \frac{M_u^{(i-1)}(x_t)}{M_n^{(i-2)}(x_t)} m_{un}^{(i-2)}(x_t) \tag{4.38}$$

Combining (4.34) and (4.38), we find a recursive expression for the beliefs:

$$\begin{aligned} M_n^{(i)}(x_t) &\propto p(y_{n,t}|x_t) \prod_{u \in G_n} \left(\frac{M_u^{(i-1)}(x_t)}{M_n^{(i-2)}(x_t)} m_{un}^{(i-2)}(x_t) \right) && (4.39) \\ &= p(y_{n,t}|x_t) \prod_{u \in G_n} m_{un}^{(i-2)}(x_t) \prod_{u \in G_n} \left(\frac{M_u^{(i-1)}(x_t)}{M_n^{(i-2)}(x_t)} \right) \\ &= M_n^{(i-2)}(x_t) \prod_{u \in G_n} \left(\frac{M_u^{(i-1)}(x_t)}{M_n^{(i-2)}(x_t)} \right). && (4.40) \end{aligned}$$

which represents novel consensus algorithm based on BP. This method is initialized by $M_n^{(1)}(x_t) = p(y_{n,t}|x_t)$. We also need to set $M_n^{(2)}(x_t)$ in order to run the algorithm defined by (4.40). Using (4.34) and (4.35), and assuming that $m_{nu}^{(1)}(x_t) = 1$, we find

$$M_n^{(2)}(x_t) = p(y_{n,t}|x_t) \prod_{u \in G_n} p(y_{u,t}|x_t) \tag{4.41}$$

As described in Chapter 3, this method guarantees convergence to $C \prod_n p(y_{n,t}|x_t)$ for cycle-free network graphs. When the network graph has cycles, the beliefs are only approximations of the true marginals (more details in Appendix B). Comparing (4.40) and (4.29), we can see that SBC is not specific instance of BP. In contrast to SBC, BP-consensus agrees on product of all local evidences (not the N_s-th root of the product), and does not rely on knowledge of $\eta_{\max}$ and N_s. We refer to this variant of DPF as DPF-BP.

Communication cost analysis

In this section, we analyze the communication cost of the three DPF methods, and compare with the cost of NCPF and CPF. We denote by N_{pack} the number of packets

that a generic node n broadcasts at a generic time t. We assume that one packet can contain P scalar values. We neglect the cost of determining ($\eta_{\max}$, $\bar{\eta}$ and N_s), so that all DPF methods will have the same communication cost.

At every iteration (except the first), nodes transmit N_w weights. In addition, nodes must perform MC, which also requires transmission of the weights in each iteration. The number of iterations of the BC is N_{it}. The number of iterations of the MC is equal to the to the diameter of the graph D_g, which represents maximum hop-distance between two nodes. Thus, the average cost of DPF per node and per time slot is

$$N_{\text{pack}}^{\text{DPF}} \approx \left\lceil \frac{N_w}{P} \right\rceil (D_g + N_{\text{it}} - 1). \tag{4.42}$$

NCPF does not require transmission of the weights, but only local data, i.e., its observations and its 2D position[9]. We denote the number of these scalar values as N_{data}. The amount of data will accumulate with iterations since the node has to transmit its own data and all received data. Since the number of iterations is equal to D_g, the cost can be approximated by:

$$N_{\text{pack}}^{\text{NCPF}} \approx \sum_{k=0}^{D_g-1} \left\lceil \frac{\bar{\eta}^k N_{\text{data}}}{P} \right\rceil, \tag{4.43}$$

where we approximate the degree of the each node with average network degree ($\bar{\eta}$).

The cost of CPF depends on many factors, including the routing protocol, and the position of the fusion center. Taking into account that in CPF each node transmits its information once (in contrast of D_g times, in NCPF), and that the fusion center is on an edge of the area, the average cost can be roughly approximated with

$$N_{\text{pack}}^{\text{CPF}} \approx \frac{N_{\text{pack}}^{\text{NCPF}}}{D_g}. \tag{4.44}$$

Note that this cost is not evenly distributed over network.

From (4.42) we see that the DPF methods are fully scalable, since increasing the number of the nodes (by increasing its density) will not affect the cost. Although beyond the scope of this chapter, we mention that if one prefers to use parametric approximations [41, 42, 44] instead of N_w messages, only parameters of the beliefs and 2D sensor positions should be transmitted, in each iteration. It is also possible to transmit only large weights (larger than predefined threshold) as in [106], or use other techniques for weight compression.

[9] If all the sensors learn the measurement model online, learned parameters also have to be transmitted.

Making the reasonable assumption that $N_{\text{it}} = D_g + 1$, we can quantify when DPF is preferred over NCPF, i.e., when $N_{\text{pack}}^{\text{DPF}} < N_{\text{pack}}^{\text{NCPF}}$:

$$\left\lceil \frac{N_w}{P} \right\rceil < \frac{1}{2D_g} \sum_{k=0}^{D_g-1} \left\lceil \frac{\bar{\eta}^k N_{\text{data}}}{P} \right\rceil. \tag{4.45}$$

This condition is important in order to avoid over-using of consensus-based methods. For example, if the network is fully-connected ($D_g = 1$), or if the packet size is sufficiently large to afford transmission of all accumulated data (i.e., $P > \bar{\eta}^{D_g-1} N_{\text{data}}$), NCPF should be applied. On the other hand, if the communication radius is very small (i.e., if D_g is very large), DPF methods should be applied. Note that a similar comparison can be done with CPF (i.e., using (4.42) and (4.44)), but note that the communication cost is not the unique reason why CPF method is not used (see Section 4.3.1).

4.3.4 Simulation results

Simulation setup and performance measures

We assume that there are $N_s = 25$ sensors semi-randomly deployed in a 100m x 100m area: the area is divided into N_s square-shaped cells, and one sensor is randomly placed in each of them. The positions of these sensors are perfectly known. There is also one target in the area which is moving with constant speed $V = 5$m/s according to a Gaussian random walk. An example of a track in a 25-node network is shown in Figure 4.10. The sampling interval is set to $T_s = 1$s, and number of these intervals is set to $N_t = 50$. We set the sensing radius to $r = 25$m, and vary the communication radius R. We assume that the measured distance is distributed according to Gaussian mixture with two components, in which one component is outlier. The parameters of this noise are set to following values: $\mu_d = (1m, 10m)$, $\sigma_d = (1m, 1m)$ and $w_d = (0.75, 0.25)$. We use $N_p = 200$ particles. The results are averaged over $N_{\text{mc}} = 100$ Monte Carlo runs.

We will compare CPF, NCPF, and the three DPF methods (DPF-SBC, DPF-BG, and DPF-BP). We consider two performance metrics: RMSE in the position error e_{rms}, and, for DPF methods, the average disagreement in the position error e_{dis}. Introducing $e_{n,t,s}$ as the target positioning error (i.e., Euclidean distance between the true and estimated position of the target) at node n, at time t in simulation run s, we have:

$$e_{\text{rms}} = \sqrt{\frac{\sum_{n,t,s} e_{n,t,s}^2}{N_s N_t N_{\text{mc}}}}, \tag{4.46}$$

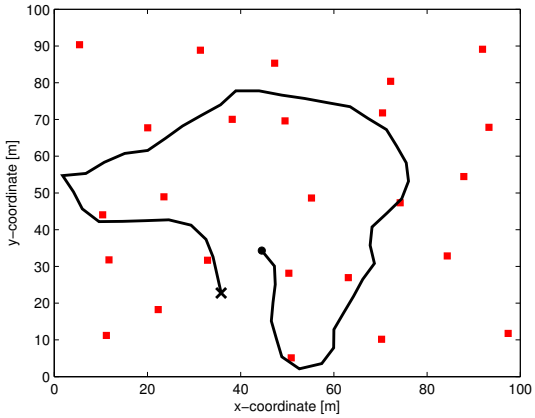


Figure 4.10: Example of track in 25-node network. Sensors are marked with red squares, the starting point of the track with a dot, and the destination point with an X.

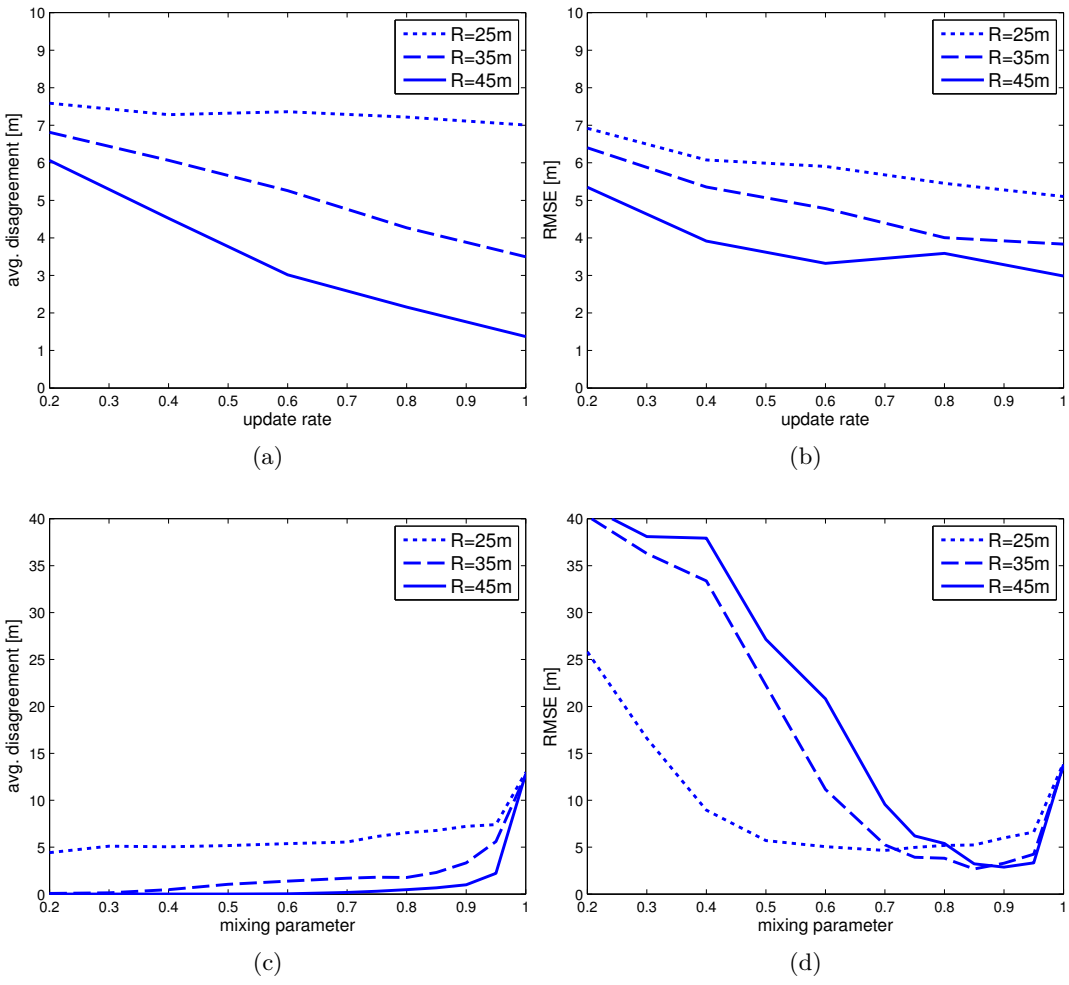


Figure 4.11: (a) Average disagreement of DPF-BC, (b) RMSE of DPF-BC, (c) Average disagreement of DPF-BG, and (d) RMSE of DPF-BG.

and

$$e_{\text{dis}} = \frac{1}{N_t N_{\text{mc}}} \sum_{s,t} (\max_n(e_{n,t,s}) - \min_n(e_{n,t,s})). \tag{4.47}$$

Determination of consensus parameters

Having defined scenarios and metrics, we perform an initial test to find reasonable values of the update rate ξ, and the mixing parameter γ. To that end, we analyze e_{rms} and e_{dis} of DPF-BC and DPF-BG w.r.t. these parameters, for different values of communication radius. The results are shown in Figure 4.11. As expected, $\xi \approx 1$ consistently provides the best performance, so the SBC exponent is set to $\epsilon = 1/\max_n(\eta_n)$. On the other hand, the best value of γ is increasing as we increase communication radius. We decided to model the optimal value of γ, as function of average node degree in the network $\bar{\eta}$, as:

$$\gamma_{\text{opt}}(\bar{\eta}) = 1 - ae^{-b\bar{\eta}}, \tag{4.48}$$

where $a = 0.49$, and $b = 0.17$ are found by fitting the training data. Note that function (4.48) is appropriate in a sense that it guarantees $0 < \gamma_{\text{opt}} < 1$ (for $0 < a < 1$, $b > 0$). It is also important to make sure that this value provides sufficiently small disagreement over the network, which is according to results increasing with γ. However, comparing results in Figure 4.11c and Figure 4.11d, we can see that γ_{opt} is a reasonable choice in terms of disagreement.

Performance results

We will first investigate the convergence as a function of the number of iterations, for $R = 25$m and $R = 45$m. From Figure 4.12, we draw a number of conclusions. First of all, CPF and NCPF provide the best RMSE performance, as they have access to all observations. Among the DPF methods, DPF-BG and DPF-SBC provide better RMSE performance than DPF-BP, as the latter algorithm is affected by the loops in the factor graph, leading to biased beliefs. On the other hand, DPF-BP offers the fastest convergence. This is expected since it is empirically known [77, 121] that BP often converges after a finite number of iterations (in our scenario, usually for $N_{\text{it}} \approx D_g + 1$). In fact, using (4.40), it is straightforward to see that $N_{\text{it}} = D_g + 1$ leads to a minimal RMSE, since then all local likelihoods are available at each node. A further increase of the number of iterations will only increase the amount of over-counting of the local likelihoods, thus leading to biased beliefs. DPF-SBC is consistently the slowest method in terms of disagreement, but it is the unique DPF method that guarantees the convergence in terms of the both metrics.

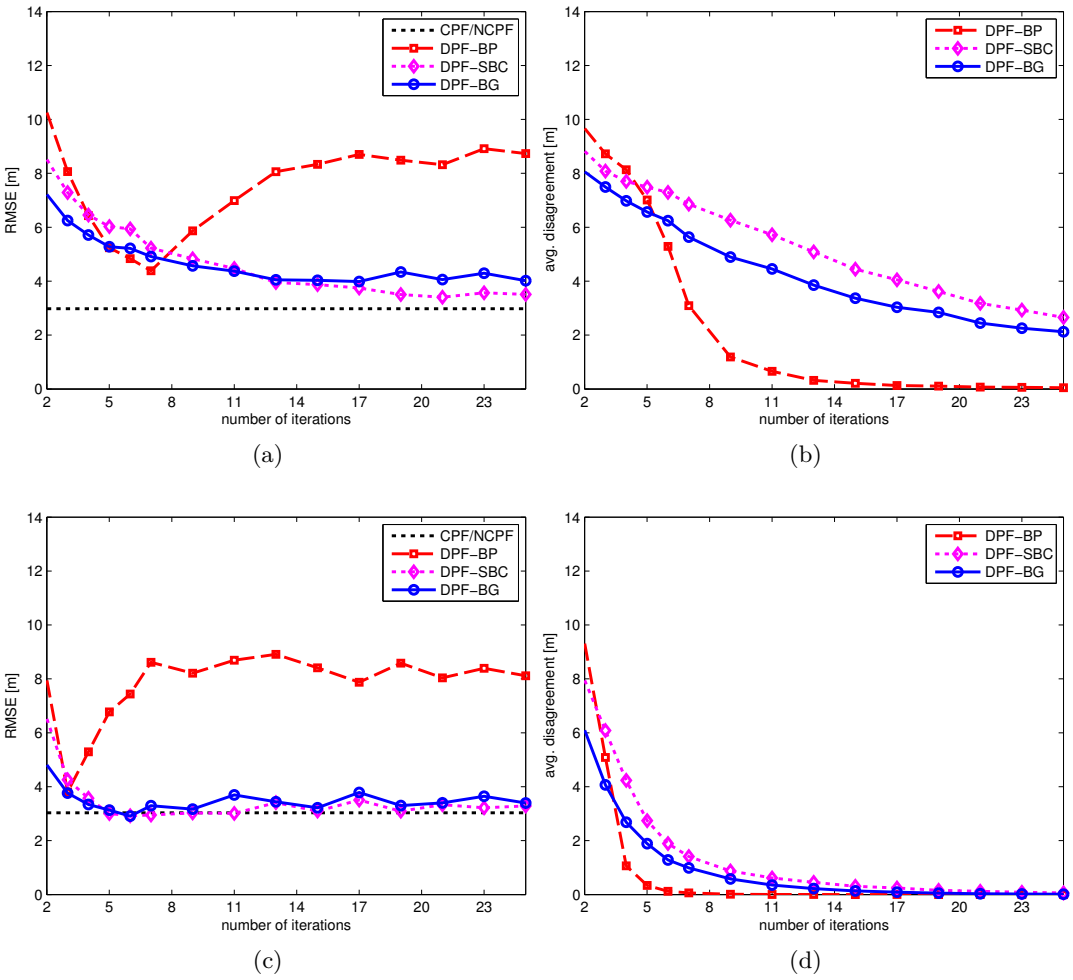
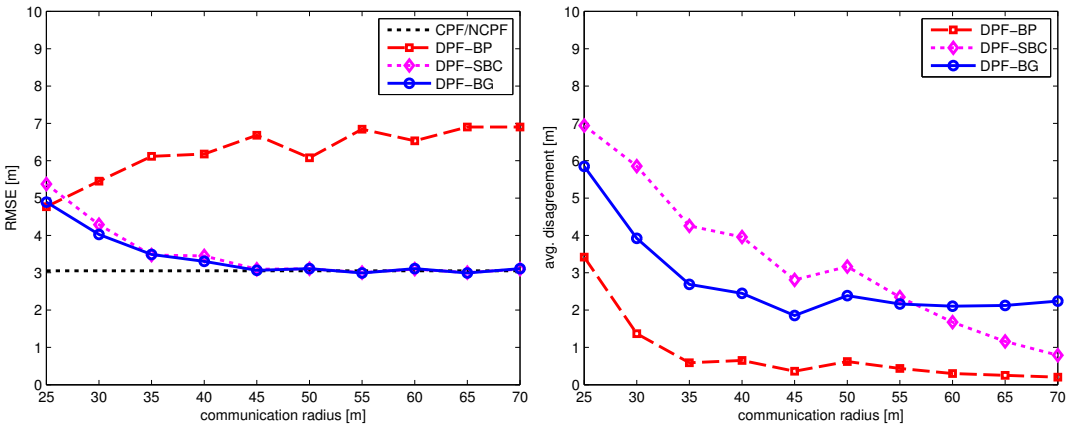


Figure 4.12: Performance comparison of DPF methods as a function of the number of iterations. (a) RMSE, $R = 25$m, (b) avg. disagreement, $R = 25$m, (c) RMSE, $R = 45$m, and (d) avg. disagreement, $R = 45$m.

Secondly, we will vary the communication radius R, and fix $N_{\text{it}} = \lceil L/R \rceil + 1$, as an approximation of $N_{\text{it}} = D_g + 1$. Here L is the diameter of the deployment area ($L = 100\sqrt{2}$ m, in our case). As we can see in Figure 4.13, DPF-BG and DPF-SBC achieve the best RMSE performance, close to the RMSE of CPF/NCPF for large R. On the other hand, DPF-BP performs the best in terms of disagreement, for all considered values of R. However, DPF-BP performs poorly in terms of RMSE. Note that if we use exactly $D_g + 1$ iterations, the performance of DPF-BP will be significantly better[10]. Of course, in practice the network will have no knowledge of D_g. Finally, we can also see that DPF-SBC provides very good agreement for large R.

[10]In our example, according to Figure 4.13a and Figures 4.12a and 4.12c, $\lceil L/R \rceil = D_g$ only for $R = 25$m.



(a) (b)

Figure 4.13: Performance comparison of DPF and CPF/NCPF as a function of communication radius, of: (a) RMSE, and (b) average disagreement.

Thirdly, we will evaluate the communication cost, and analyze the average number of packets per node as a function of the communication radius R for $N_{\text{it}} = \lceil L/R \rceil + 1$. We consider networks with 25 and 100 nodes, and packets sizes of $P = N_w$, and $P = 5N_w$, where $N_w = N_p = 200$, and $N_{\text{data}} = 3$. As we can see in Figure 4.14, DPF-based methods provide nearly constant communication cost as a function of R, since (4.42) only depends linearly on D_g. By comparing Figures 4.14a and 4.14b, and Figures 4.14c and 4.14d, we can see that for DPF-based methods the communication cost does not depend on N_s. Thus, these methods are fully scalable. On the other hand, the communication cost of CPF/NCPF is highly sensitive to R and N_s. It increases as R increases (while $\hat{D}_g$ is fixed), and decreases significantly when $\hat{D}_g$ decrements its value (e.g., for $R = 50\sqrt{2}$). Overall, decreasing $\hat{D}_g$ has the largest effect (see (4.43)), so the total cost has decreasing tendency with R. In addition, since the increased N_s affects $\bar{\eta}$, the communication cost will be significantly larger. Regarding the effect of P, we can see that larger values of P will make CPF/NCPF cheaper, as more data can be aggregated in one packet. Finally, comparing with NCPF, we can see that DPF methods have a lower communication cost for $R < 70$m, except when P is very large (as in Figure 4.14c).

Finally, even without numerical comparison, we can claim that other asynchronous tracking methods based on average consensus [41, 42, 44] perform worse than DPF-SBC due to the likelihood compression (parametric approximation), and that methods based on randomized gossip [74, 106] perform significantly slower than method based on BG (DPF-BG).

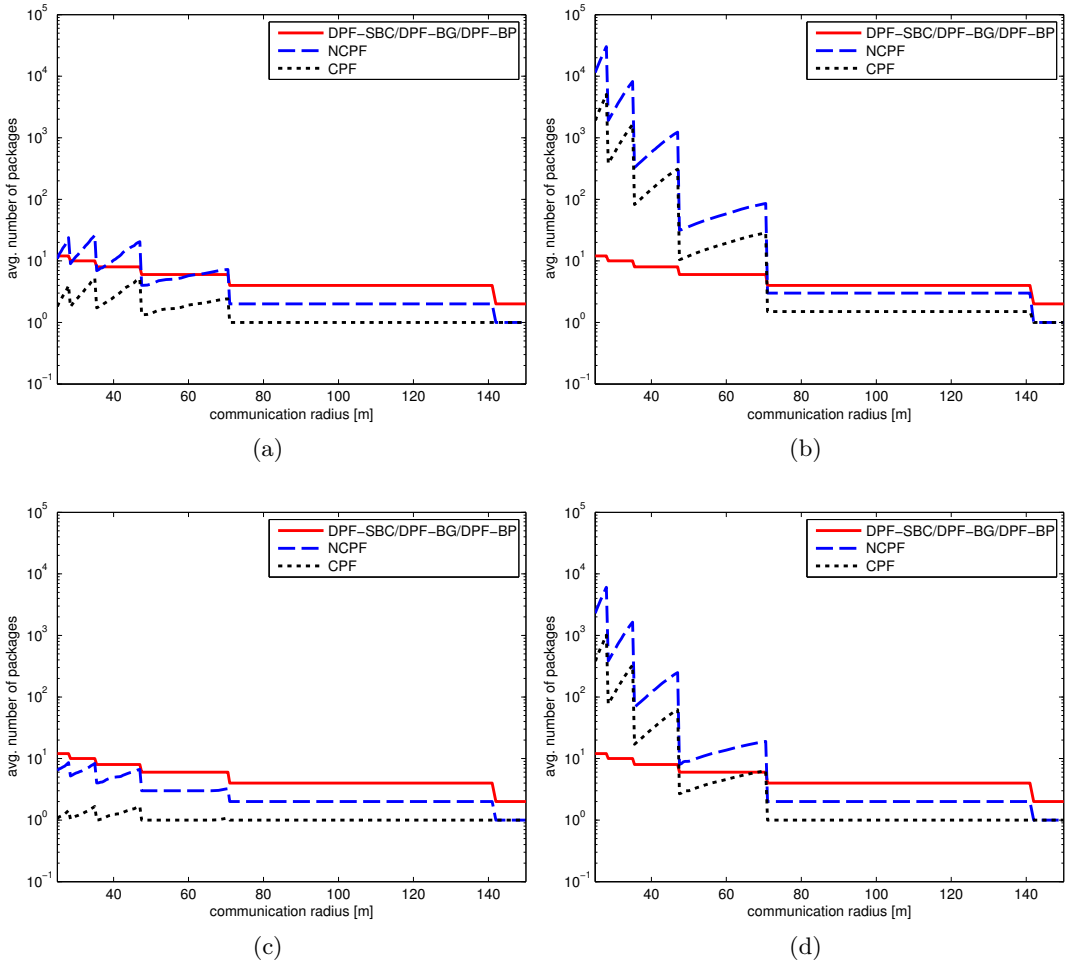


Figure 4.14: Communication cost comparison as a function of the communication radius, for: (a) 25-node network, $P = N_w$, (b) 100-node network , $P = N_w$, (c) 25-node network, $P = 5N_w$, and (d) 100-node network, $P = 5N_w$.

4.4 Summary

In this chapter, we proposed novel algorithms for two different mobile scenarios: cooperative localization in mobile networks, and distributed target tracking. For cooperative localization in mobile networks, we proposed different variants of NBP method. Our main contributions are: i) an optional 1-leg smoothing done almost in real-time, ii) a novel low-cost communication protocol, and iii) more efficient sampling techniques. For distributed tracking, we proposed three DPF variants based on BC algorithms (DPF-SBC, DPF-BG, and DPF-BP). In contrast to previous methods, these methods can approximate global likelihood function in non-parametric form. According to our results, DPF-BG should be used in all tracking applications where minimal expected error is crucial. On the other hand, if the

agreement of the estimates in the networks is more important than absolute error, DPF-BP could be a good choice. DPF-SBC, which guarantees convergence after large number of iterations, could be applied when the cost and latency are not crucial.

Chapter 5

Experimental study of cooperative localization and tracking methods

5.1 Introduction

The main goal of this chapter is to provide the experimental analysis of some of the proposed algorithms in previous chapters. In particular, we analyse cooperative localization based on NBP, and NBP-ST, described in Chapter 3, and distributed tracking using DPF based on belief consensus algorithms, described in Chapter 4. We also provide experimental analysis of centralized single-node localization and tracking problems. Experimental analysis of NBP and NBP-ST cooperative localization methods [87,88] is performed using RSS data obtained in indoor environment. For these experiments, Crossbow's IRIS wireless motes has been used, which is fully compatible with ZigBee/IEEE802.15.4 standard. Distributed tracking has been analysed using novel semi-passive RFID system, proposed in [31]. This system is composed of a standard Ultra High Frequency (UHF), ISO-18006C compliant RFID reader, a large set of standard passive RFID tags whose locations are known, and a newly developed tag-like RFID component that is attached to the items that need to be localized. The new semi-passive component, referred to as *sensatag* (sense-a-tag), has a dual functionality wherein it can sense the communication between the reader and standard tags which are in its proximity, and also communicate with the reader like standard tags using backscatter modulation. Based on the information conveyed by the sensatags to the reader, range-free localization and tracking algorithms can be developed. We implement and test centralized localization [4] and tracking [94] algorithms in indoor environment. Finally, since current hardware is

not appropriate for distributed applications, we reuse the real model (used for the centralized algorithm) to simulate distributed tracking algorithms.

5.2 Experimental study of NBP and NBP-ST methods

We start with the description of the setup used for the experiments performed in our lab. We then describe the reliable indoor model created using obtained measurements and import all data into Matlab in order to check the performance of the NBP and NBP-ST methods (described in Chapter 3) in high-density WSN.

5.2.1 Experimental setup

For our experiments, we use Crossbow's IRIS motes [25] (Figure 5.1a), ultra low-power wireless devices with long-range communication, fully supported under TinyOS operating system. They are equipped with AT86RF230 transceiver and ATMega 1281 microcontroller. ATMega 1281 [5] is low-power Atmel 8-bit RISC-based microcontroller with 128KB flash memory, 8KB SRAM, and 4KB EEPROM. The device achieves a throughput approaching 1 MIPS per MHz, balancing power consumption and processing speed. AT86RF230 [3] is high-performance RF-CMOS 2.4 GHz radio transceiver, fully compatible with ZigBee/IEEE802.15.4 standard. The transmitter provides programmable output power: -17 dBm up to 3 dBm. It is specifically designed for low cost applications such as wireless sensor networks, PC peripherals, consumer electronics and industrial control, sensing and automation. Its power consumption is 16.5 mA at maximum transmit power (3 dBm), and 20 nA in sleep mode. The receiver, with -101 dBm sensitivity, generates digital signal with 3 dB granularity. The data is stored in a 128-byte dual port SRAM, from which 8 bytes are reserved.

In order to estimate the distance between sensors, we placed two sensors, 2m above the floor, in our 5m x 10m lab (Figure 5.1b) and set the transmission power to 3 dBm. There are no obstacles between sensors, but the RSS is affected due to the multipath components and other devices in vicinity. We obtained RSS measurements at 8 equidistant inter-sensor distances ($k \cdot 1.2$m, $k = 1, ..., 8$). For each of them, we obtained 1000 measurements. Because of the 3 dB granularity of RSS, we assume that the real power is a random variable uniformly distributed within the interval (RSS - 1.5 dB, RSS + 1.5 dB).

(a)

(b)

Figure 5.1: (a) Crossbow's IRIS wireless sensor node, (b) Illustration of the experiment in our lab.

5.2.2 Indoor modeling using RSS measurements

Using obtained RSS measurements, our goal is to estimate all necessary parameters and estimates for application of NBP/NBP-ST in indoor environment: path-loss exponent, distance estimates, probability of detection and potential functions.

Path-loss exponent estimation

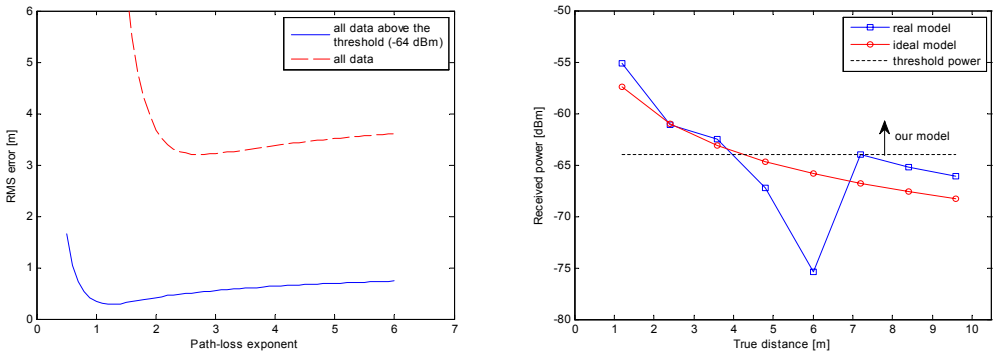
We first define a reference point ($P_0(d_0 = 2.4m) = -61$ dBm). The path-loss exponent (n_p) could be easily obtained using another reference point, but this is not a good solution. Thus, we use an alternative approach in which all the measured data has been used to find n_p, by minimizing the RMSE:

$$e^d_{rms}(n_p) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d^i_{measured}(n_p) - d^i_{true})^2} \tag{5.1}$$

where n is the number of inter-sensor distances (in our case, $n = 8$) and $d^i_{measured}(n_p)$ is, according to Section 2.2.1 (in Chapter 2), given by:

$$d^i_{measured}(n_p) = d_0 \cdot 10^{-\frac{P^i_r[dBm] - P_0[dBm]}{10 n_p}} \tag{5.2}$$

We find the optimal value of n_p graphically, which is, according to Figure 5.2a (dashed line), equal to 2.7.



(a) (b)

Figure 5.2: Illustration of (a) path-loss exponent estimation, and (b) reliable model for distance estimation

Distance estimation

Using obtained measurements and estimated n_p, we can estimate the distance. As we expected, our indoor model is not similar to the ideal one (Figure 5.2b), so the distance cannot be always trustfully estimated using (5.2). For example, the averaged received power of -66 dBm corresponds to three different distances (4.6m, 7m and 9.6m), so the mote can just guess. This is because the power is not monotonically decreasing function of the distance. Therefore, we have to cut out the area below the threshold power (-64 dBm) because this area corresponds to the non-monotonic part of the function. Above the threshold, each received power corresponds to the unique distance, which makes this model reliable for our scenario. In addition, since we excluded the data below the threshold, we must re-estimate n_p using only the remaining data. According to Figure 5.2a, $n_p = 1.2$. We illustrate in Figure 5.3, the distance estimation which corresponds to the true value of 1.2m. As we can see, the error distribution ($d_{measured} - d_{true}$) is not similar to the log-normal distribution, so we will use nonparametric form of the error distribution. Moreover, we have three different sets of error samples (for 1.2m, 2.4m, and 3.6m). Thus, in order to import these samples into Matlab, we will simply draw the sample from the nearest error distribution, and then add it to the true distance (i.e., this is nearest neighbor interpolation, so for the true value of e.g., 2.9m, we use the error sample for 2.4m).

Model for probability of detection

For each inter-sensor distance, we found that RSS is above the power defined by sensitivity (Figure 5.4). This is expected because we set the transmission power to

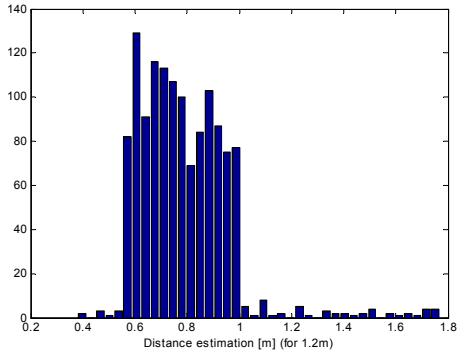


Figure 5.3: Histogram of distance estimate which corresponds to the true value of 1.2m

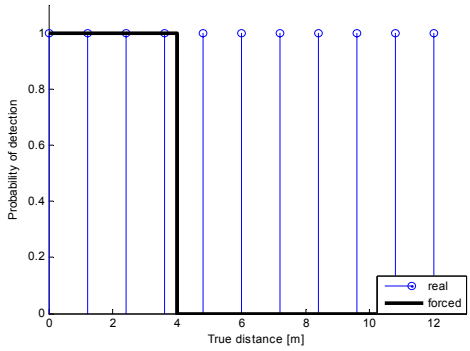
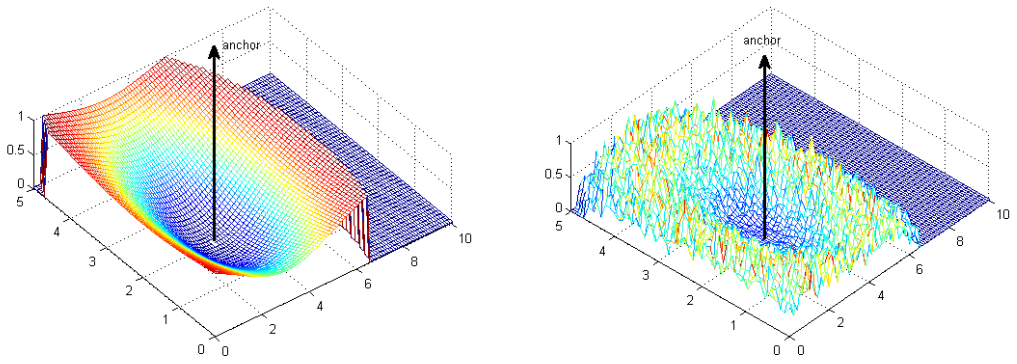


Figure 5.4: Estimated probability of detection

the maximum which could even provide us about 75m radius, according to ZigBee standard. Anyway, we have to follow defined reliable model, so we assume that, if the power is less than threshold (-64 dBm), there is no communication between nodes. This could be easily forced by software. As we can see in Figure 5.2b, the corresponding distance is 4m, so this will be the maximum value of transmission radius. Note that, in our case, we did not detect communication failures (link quality indicator is always maximum), so we set $P_d = 1$ in the transmission range. This is expected due to the very small distance between nodes.

Model for potential functions

We have to define the single-node and the pairwise potential function. Since we do not have any a priori information about positions of unknown nodes, single-node potential of unknown node is equal to 1 in the area defined by Figure 3.4. Regarding pairwise potential, according to Section 3.3.1 (in Chapter 3), given anchor node (or particle of unknown node), the position of other node is shifted in the random



(a) (b)

Figure 5.5: Pairwise potential function $\psi_{ut}(x_t^*, x_u)$ (x_t^* - anchor, x_u - unknown) using (a) log-normal model, and (b) empirical model from our lab.

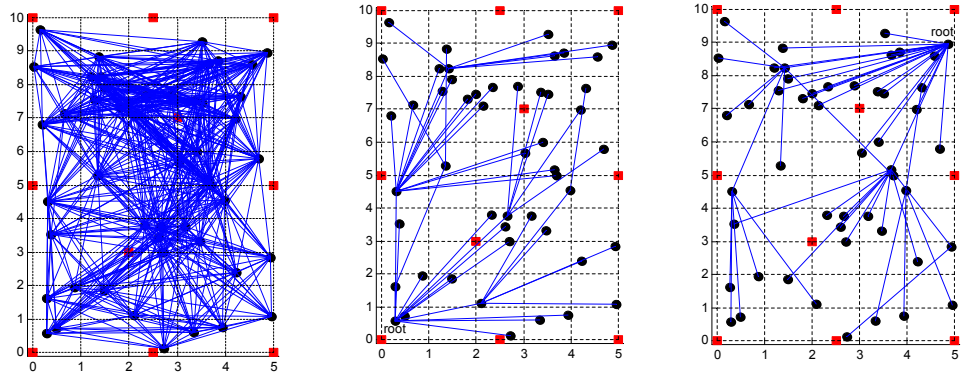
direction by measured distance between nodes. We obtained density function using a spherically symmetric Gaussian kernel [104]. We illustrate theoretical (log-normal) model in Figure 5.5a, and our indoor model in Figure 5.5b.

5.2.3 Simulation results

We assume that there are 50 sensors on unknown position and 10 anchors in 5m x 10m area as shown in Figure 5.6a. The unknown nodes are deployed randomly within this area and anchor nodes are fixed (8 along the edges an 2 in center area). We use $N = 50$ particles, and vary transmission radius (R = 2m - 4m). The number of iteration is set to $N_{iter} = 3$, which is sufficient, taking into account maximum value of R. We import our distance model (see previous section), obtained using RSS measurements. The results are averaged over 30 Monte Carlo runs.

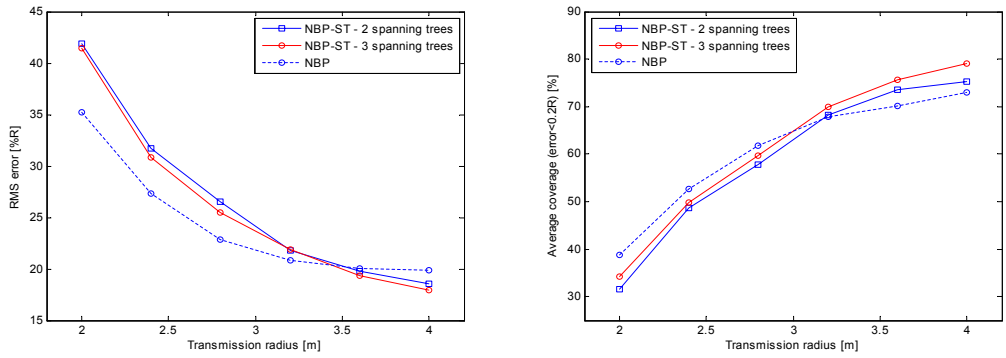
Using the defined scenario, we compared NBP and NBP-ST algorithms. For NBP-ST, we used 2 and 3 STs (see Figure 5.6b and Figure 5.6c). As we can see in Figure 5.7, NBP-ST performs better than NBP starting from some value of R ($R = 3.5$m in this case), in terms of RMSE error and coverage. To measure the communication cost, we count elementary messages, as in previous chapters. However, in this case, we assume that this data is represented in single precision floating-point format that occupies 4 bytes in the memory. Since 8 bytes are already reserved (see Section 5.2.1), the size of elementary message is 12 bytes. According to Figure 5.8b, NBP-ST performs better than NBP for $R > 3.3$m only if we use 2 spanning trees. Regarding computational cost[1] (Figure 5.8a), NBP outperforms NBP-ST. If we keep

[1]Note that we show the joint computational cost of both spanning tree formation and NBP method.



(a) (b) (c)

Figure 5.6: (a) Original network, (b),(c) two corresponding STs. Connections between anchors (marked with red squares) and unknowns (marked with black circles) are not shown



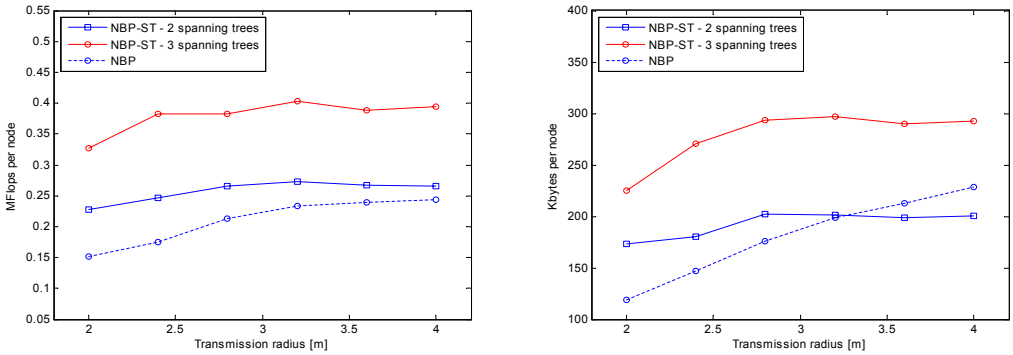
(a) (b)

Figure 5.7: Comparison of (a) accuracy, and (b) coverage.

increasing R ($R > 4$m), we can outperform NBP, but RSS measurements are not reliable in this case[2]. Finally, if we use 3 or more spanning trees, both computational and communication cost will be obviously significantly higher.

The main conclusion is that NBP-ST (with 2 STs) performs better than NBP in terms of accuracy and communication cost, for $R > R_{min}$. This conclusion is the same as for results based on the theoretical data (see Section 3.6.2 in Chapter 3). However, there are two important differences: i) we did not achieve smaller computational cost (caused by bounded R), and ii) the error level is slightly larger (caused by indoor environment).

[2]Larger R can be likely used with TOA measurements which are usually more accurate [76].



(a) (b)

Figure 5.8: Comparison of (a) computational cost, and (b) communication cost.

5.3 Localization and tracking using novel RFID system

RFID is a well-known technology for real-time identification of various assets and users. One of the main goals of RFID technology is to enable ubiquitous asset visibility. Precise determination of an asset location is of great importance in achieving this goal. Accurate localization and tracking using RFID can enable several applications such as location of tagged items in warehouses and location of assets and personnel in hospitals and offices [86, 98]. State-of-the-art RFID localization methods can be broadly classified into three categories [14]: i) distance-based methods, ii) methods based on scene-analysis (fingerprinting), and iii) proximity-based (or range-free) methods.

The main problem with the distance-based and scene-analysis methods is that they are affected by dynamic changes in the environment. One direction of investigation for resolving this problem is to work with *proximity-based* methods that exploit binary information, i.e., information about a target being within the ranges of the reference tags. The location estimate is found either by associating the location of the target with that of the closest reference tag, or as the centroid obtained from the locations of all the reference tags that detected the target. This type of methods has already been used for localization of mobile robots or sensors with a large number of references with known positions [30, 53, 68].

Location of assets is a problem that was traditionally solved with active RFID and WiFi based technologies. These solutions require placing an active tag on assets and deploying a dense WiFi access point infrastructure. They become economically infeasible when tagging large number of low to medium cost items that are densely collocated. Estimating the location of an asset tagged with a standard passive or

semi-passive tag has been a much sought after application since the inception of RFID. There are some solutions based on technologies that exist on the market. Although they are standard compliant, these solutions do not rely on off-the-shelf UHF readers and require directional antennas. Therefore, we use a novel type of semi-passive UHF RFID tag that has the capability to detect and decode backscatter signals from RFID tags in its proximity and to communicate this information to a standard RFID reader. We refer to this tag as *sensatag* (from sense-a-tag) [4, 31]. We performed many experiments in the laboratory[3] so that we could quantify the performance of this system for localization and tracking. In addition, we also imported measurements into Matlab in order to test distributed tracking algorithms from Chapter 4.

5.3.1 A novel sensatag-based RFID system

Passive and semi-passive UHF RFID tags do not have on-board radios. They communicate with the reader using the principle of backscatter modulation wherein, the reflection cross section (RCS) of the tag antenna is varied in accordance with the data to be conveyed to the reader [33]. This modulates the signal reflected from the tag antenna to the reader. The tag backscatter is a weak signal that is further affected by multipath reflections and other ambient interferences in cluttered indoor environments like warehouses, retail stores, libraries, and offices [11]. This results in a low signal-to-noise ratio for the tag response received by the reader. Hence conventional location techniques based on the measurement of some characteristic of the tag's response like RSS, TOA, or TDOA become highly inaccurate and unreliable for localization with passive and semi-passive RFID systems. For example, RFID system, that we use for experiments, provides very large uncertainty in RSS in indoor environment (see Figure 5.9), so it is not suitable for distance estimation.

Our approach to localization is based on the addition of a new component to a standard RFID system (with one reader, number of passive tags, and host computer; see Figure 5.10), called *sensatag* [31]. Sensatag is semi-passive, tag-like component that has the following capabilities: i) to detect and decode backscatter signals from RFID tags in its proximity and ii) to communicate with the reader using backscatter modulation. On top of these basic capabilities, the sensatag has incorporated, a novel *locator protocol*, which is fully compatible with the EPC Global Class 1 Gen 2 standard (ISO-18006C). This protocol enables the sensatag to communicate with a standard reader and convey binary information about the presence or absence of a responding tag in its proximity. Figure 5.11 illustrates the sensing zone of the

[3]The experiments are performed at Center of Excellence of Wireless and Information Technology (CEWIT), Stony Brook University, NY, USA.

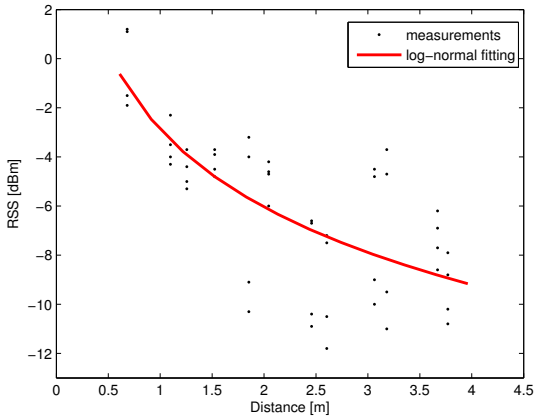


Figure 5.9: Received Signal Strength (RSS) (from reader) at sensatag as function of distance.

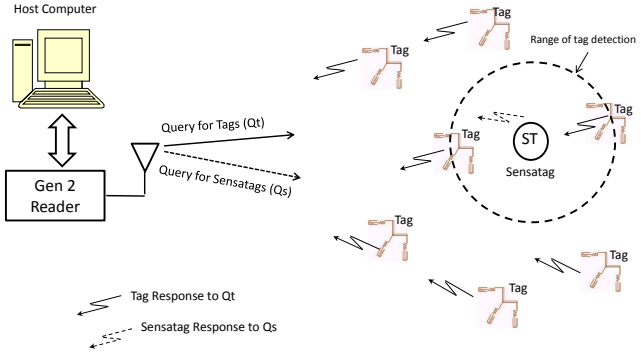


Figure 5.10: Architecture of the sensatag-based RFID system.

sensatag for two different positions of the reader.

The sensatag communicates passively without an on board radio. An on board battery is used for powering up the sensatag circuitry. Thus, in its current form, the sensatag is a semi-passive device. In Appendix C.1, we briefly describe the various functional blocks that make up the sensatag.

Locator Protocol

The sensatag implements a novel *locator protocol* which enables it to convey binary association information about tags in its vicinity to a standard reader. In order to implement this functionality, the locator protocol specifies two states of operation for the sensatag. In the first state or the *listen* state, the sensatag listens for backscattering tags in its vicinity. In the second state or the *respond* state, the sensatag itself

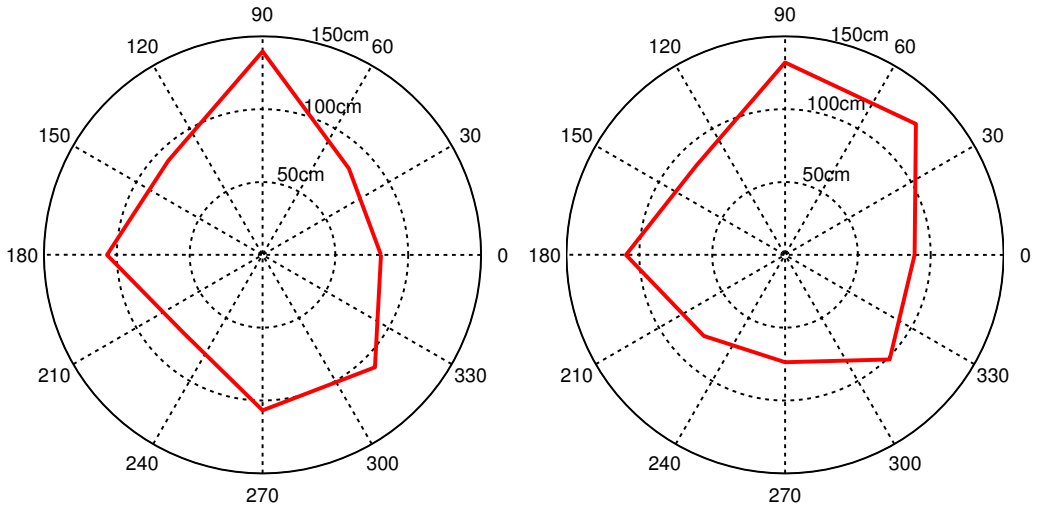


Figure 5.11: Sensing zone of the sensatag which is: (a) 1.8m, (b) 3.6m, away from the reader. In the area outside the polygon marked with the red line, the probability of detection is zero.

functions as an RFID tag and conveys the information of the tags detected when it was in the *listen* state as part of its EPC ID payload. The transition between the two states is done based on different types of queries (Q_t - tag query and Q_s sensatag query) received from the reader using the *Select* functionality provided by the Gen 2 standard. In the query round Q_t the sensatag acts as a sensor detecting, decoding and storing information about the responding tags within its vicinity. In the subsequent Q_s query round, the sensatag conveys the binary tag association, along with its own unique identifier information to the reader using backscatter modulation. The localization or tracking algorithm running on the reader side aggregates the binary association information from successive query rounds and determines the location of the sensatag with respect to the pre-deployed tags in the environment. The whole protocol is summarized in Table 5.1.

5.3.2 Sensatag localization

In the system described in previous section, passive RFID tags are deployed at predefined locations within the environment where localization is to be performed. A sensatag is attached to the target of interest. The reader is programmed to send out alternating queries for the tags and sensatags using the *Select* functionality. The sensatag attached to the target operates using the locator protocol described above and conveys binary information about presence or absence of responding tags to the reader.

Table 5.1: Functions of the host and the sensatag during different phases

	Host	Sensatag
Listening (Q_t)	- Initiate query Q_t - Obtain tags' IDs	- Listening to the reader - Detecting the tag - Storing tag's ID
Reporting (Q_s)	- Initiate query Q_s - Obtaining sensatag's data	- Listening to the reader - Backscattering its ID and partial IDs of detected tags.
Processing	- Associating tags with sensatags - Localization algorithm - Tracking Algorithm	

Let us assume that we have M reference (passive) tags with known 2D positions, x_i $(i = 1, 2, \cdots, M)$ and one sensatag with unknown position l. A reference tag can be detected by a sensatag with probability p_i. This probability depends on various factors, but primarily on the distance between the reference tag and the sensatag, orientation, and the power of the reader. This probability is easily estimated by counting the number of detections of a tag by a sensatag in a fixed number of reader queries.

Our main goal is good performance (without calibration) in environments with dynamical changes, so we decided to use three simple localization methods that should work well in such circumstances. They are based on i) association, ii) centroids, and iii) weighted centroids.

With association we simply associate the sensatag with the nearest passive tag. The proximity is measured by comparing the p_is of each reference tag. The main drawback of association is when more passive tags are detected by the sensatag, the p_is may not correctly reflect the distance from the sensatag, which will imply that the sensatag will be associated with a wrong passive tag. As a result, the position error will be larger.

One simple way of building a more robust method is to implement averaging of the positions of all the passive tags that have been detected by the sensatag. In that case, the position of the sensatag is computed by: $\widehat{l} = \sum_i x_i/N$, where the summation is over the locations of the tags that have been detected and N is the total number of detected tags by the sensatags. Therefore, the estimated position will be the *centroid* of the positions of the detected passive tags. This approach does not take into account the number of detections.

A natural extension of the centroid method, is the weighted centroid (WC), where the estimated position is the weighted average of the positions of the detected

tags. Since it is expected that the closer tags will be detected more times than more distant ones, the weights are proportional to probabilities of detection of the tags. So, the estimated position is found as:

$$\widehat{l} = \sum_i p_i x_i = \sum_i \frac{N_d^i}{N_q} x_i \tag{5.3}$$

where N_q is number of queries, and N_d^i number of detections of a tag i by a sensatag.

A description of two potential applications (shelf identification and direction of movement estimation) is provided in Appendix C.2.

5.3.3 Sensatag tracking

Let us assume that we have K reference (passive) tags with known 2D positions, l_k ($k = 1, 2, \cdots, K$) and one sensatag attached to an object with an unknown position and velocity x_t at time t. A reference tag can be detected by an sensatag with probability $p_{k,t}$. This probability depends on various factors, but primarily on the distance between the reference tag and the sensatag, orientation, and the power of the reader [11]. This probability is easily estimated by counting the number of detections of a tag by an sensatag in a fixed number of reader queries. Using this observation, our goal is to estimate x_t at each time t.

We use the following discrete state-space model:

$$x_{t+1} = Ax_t + Bu_t \tag{5.4}$$

$$y_t = Cx_t + v_t \tag{5.5}$$

where $x_t = [x_{1,t}\ x_{2,t}\ \dot{x}_{1,t}\ \dot{x}_{2,t}]^T$ is the state vector at time t, which includes the position and velocity of the sensatag that we want to estimate, $u_t = [u_{1,t}\, u_{2,t}]^T$ is the process noise (which accounts for the variation of the speed), $y_t = [y_{1,t}\ y_{2,t}]^T$ is observation at time t, and $v_t = [v_{1,t}\, v_{2,t}]^T$ is observation noise. The observation is given as $y_t = \sum_k p_{k,t} l_k$, i.e., the weighted average of the positions of the detected tags (i.e., WC method described above). This position estimate is more accurate (see next section) than other estimates found either by non-weighted average, or simply by association with the nearest reference point. It is also worth noting that, since our observations represent static position estimates, our model is linear (in contrast to distance-based method, in Section 4.3.2). Given this observation, the sampling period T_S, and assuming random motion of the target, we can define the matrices A, B, and C as follows:

$$A = \begin{bmatrix} \boldsymbol{I}_2 & T_S\boldsymbol{I}_2 \\ \mathbf{0}_2 & \boldsymbol{I}_2 \end{bmatrix},\ B = \begin{bmatrix} \frac{T_S^2}{2}\boldsymbol{I}_2 \\ T_S\boldsymbol{I}_2 \end{bmatrix}, C = \begin{bmatrix} \boldsymbol{I}_2 & \mathbf{0}_2 \end{bmatrix} \tag{5.6}$$

We apply the Bayesian approach to solve this tracking problem. filtering. At time t, our goal is to estimate the posterior distribution $p(x_t|y_{1:t})$ given the prior $p(x_{t-1}|y_{1:t-1})$ (initially, $p(x_0|y_0) = p(x_0)$ is available), the state evolution $p(x_t|x_{t-1})$ (defined by the motion model (5.4)), and the likelihood function $p(y_t|x_t)$ (defined by the measurement model (5.5)). This posterior can be found by the prediction and filtering equations [2] already used in Chapter 4 (equations (4.24), and (4.25)).

A standard closed-form solution can be found using traditional KF [116], assuming that the model is linear (as in our case), and that u_t and v_t are drawn from Gaussian distributions. The estimation of the process noise u_t is generally very difficult (it requires an accelerometer or a similar device attached to the target). Thus, we approximate u_t by a Gaussian distribution. To make the process reliable, we need to find an upper bound of the true noise, e.g., by injecting enough uncertainty into the covariance matrix. However, the measurement noise v_t can be easily obtained using real samples. Generally, we cannot expect (especially, in indoor environment) that this noise v_t is Gaussian, so KF is not an optimal solution for our problem.

Therefore, we apply the PF method [2] in which we represent the posterior PDF by a set of random samples (particles) with associated weights. We apply the well-known SIR method (called PF-SIR). In this method, the particles are drawn from $p(x_t|x_{t-1})$, then weighted by the likelihood function, $p(y_t|x_t)$, and finally, resampled in order to avoid the *degeneracy problem* (the situation in which all but one particle have negligible weights). The PF-SIR method is summarized in Chapter 4 (Alg. 11).

Note that in generally we don have a parametric form of likelihood function, so we can find its approximation using a KDE [104]. Namely, given a set of N_i calibration samples $v_t^i = y_t^i - Cx_t^i$, we have:

$$p(v_t) = \sum_i \mathcal{K}_h(v_t - v_t^i) \tag{5.7}$$

where $\mathcal{K}_h$ is the commonly used spherically symmetric Gaussian kernel: $\mathcal{K}_h(x) = \mathcal{N}(x, 0, hI)$, and h is the bandwidth which controls the variance. To find h, we use the *generalized cross entropy* estimator [13], which provides very accurate estimates. This kernel can be found offline prior to tracking. However, if the RFID system is fast enough to provide N_i samples during the sampling period (T_S) and also to compute (5.7), the likelihood function can be obtained online, at each time frame.

We can see that the main drawback of this method is high complexity. In the following text, we propose PF with different model of the likelihood function.

Improved PF method (PF-BIN)

We propose a model for the number of detections of a tag by an sensatag and show how we proceed with particle filtering. First, we model the probability of detection of a tag by an sensatag according to

$$p = \frac{1}{1+e^{\alpha(d-d_0)}} \tag{5.8}$$

where d is the distance between a tag and the sensatag, and $\alpha > 0, d_0 > 0$ are parameters of the model, with d_0 being the distance at which the probability of detection is equal to $1/2$, and α being the parameter which determines the steepness of the function.

Our measurements represent the number of times a tag is detected by an sensatag in N query rounds. We assume that during the N query rounds, the location of the object with the sensatag has not changed much (recall that the object with the attached sensatag is moving). Let the number of detections of the kth tag be equal to n_k. Then the probability of n_k is modeled by the binomial distribution, i.e.,

$$P(n_k) = \binom{N}{n_k} p_k^{n_k} (1-p_k)^{N-n_k} \tag{5.9}$$

where p_k is given by (5.8) with d replaced by d_k, the distance between the sensatag and the kth tag. In the field, there are total of K tags and for each of them we have a number of detections $n_k \in \{0, 1, \cdots, N\}$, $k = 1, 2, \cdots, K$.

Under the assumption that the parameters of the model in (5.8) are known (they are estimated offline), we proceed with particle filtering as follows (note that we also assume that at time $t-1$ we have the set of particles $x_{t-1}^{(m)}$):

Step 1: Propagate the particles by using the prior, that is,

$$x_t^{(m)} \sim p(x_t | x_{t-1}^{(m)}). \tag{5.10}$$

Step 2: Compute the likelihood of the particles $x_t^{(m)}$ given the measurements $y_t = [n_{1,t}\ n_{2,t}\ \cdots n_{K,t}]^\top$. The likelihood function is given by

$$p(y_t | x_t^{(m)}) = \prod_{k=1}^{K} \binom{N}{n_k} p_{k,t}^{(m)^{n_{k,t}}} (1-p_{k,t}^{(m)})^{(N-n_{k,t})} \tag{5.11}$$

where

$$p_{k,t}^{(m)} = \frac{1}{1+e^{\alpha(d_{k,t}^{(m)}-d_0)}} \tag{5.12}$$

and $d_{k,t}^{(m)}$ is the distance between the sensatag (whose location is defined by the particle $x_t^{(m)}$) and the k-th tag at time t. We note that the weights of the particles are

$$w_t^{(m)} \propto p(y_t|x_t^{(m)}). \tag{5.13}$$

Step 3: Resample with replacement.

We refer to this novel variant of PF-SIR, as PF-BIN method.

Distributed tracking

Current sensatag-based RFID system can only work in centralized fashion (i.e., the host computer collects the data and runs the algorithms). However, with small modifications, sensatags can also perform the localization and the tracking algorithms in distributed way (recall that they are equipped with FPGA). It is only necessary to update sensatag software, and to provide appropriate protocol for communication with neighboring sensatags[4]. Note that, in contrast to the centralized case, the sensatags must be used as reference nodes (anchors).

Taking this into account, we can use the same model from the centralized scenario for the simulation of the distributed tracking algorithms. Therefore, we can test the DPF methods (DPF-SBC, DPF-BG and DPF-BP) proposed in Section 4.3.2.

5.3.4 Experimental results

We now provide details of the experiments for studying sensatag-based RFID system for localization, tracking and related applications.

Sensatag localization

We deployed 12 passive tags in 6 reference points, where at each point we deployed two passive tags.[5] The overall area was 1.6m x 1.3m. The setup is shown in Figure

[4]If we want to increase the communication range, they should be active as well (i.e., with on-board radio).

[5]The reason for having two passive tags at (nearly) same location was to prevent missing a tag by the sensatag because of eventual destructive superposition of the signals from the reader and the

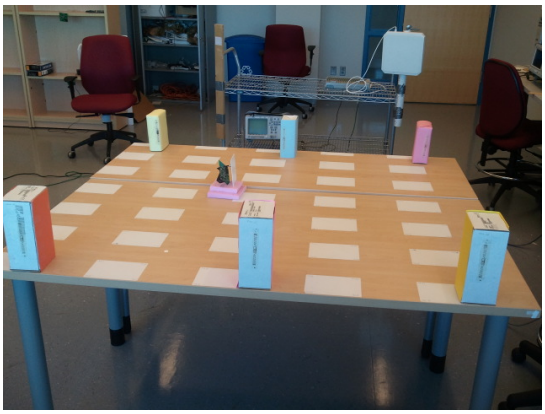


Figure 5.12: Experimental setup for sensatag localization: There are 6 reference points (shown by the standing boxes) each represented by two passive tags. There is one sensatag located in the middle. The reader and the antenna are in the background.

5.12. The difference between the reader antenna and the center of the plane in which the sensatag is placed is $1.8m$. The sensatag was placed somewhere inside the area of interest. The objective was to estimate its position in the area.

In the first set of experiments, we studied the accuracy of the estimate as a function of the reader power. We carried out localization of the sensatag at 10 different positions and computed the average error (defined as the Euclidean distance between the true and the estimated position) as a function of power. The results are shown in Figure 5.13, where we see that the association method has the worst performance but is almost constant in the studied range of reader powers. The method based on the weighted centroid outperformed the one that uses the centroid. For both methods the performance improved with increasing of the reader power. The best performance of all the methods was by the weighted centroid with a reader power of 28 dBm (the accuracy was about 14cm).

In the next experiment, we studied the effect of the distance between the sensatag and the passive tags on the probability of detection. To that end, we acquired 20 independent measurements at 20 grid points. The results are shown in Figure 5.14. We can see that the probability of detection can vary considerably even for the same distance. We, however, expect this variability; it is due to the different multipath components and other factors that play role in formation of the signal received by the sensatag. We fitted the data with four-degree polynomial function, which is also shown in the Figure 5.14. The curve shows how the probability of detection decreases monotonically with distance. The decrease of the probability of detection with distance is the main motivation for using the weighted centroid

tag. However, in order to avoid an interference between these two tags, they should be minimum 10cm from each other.

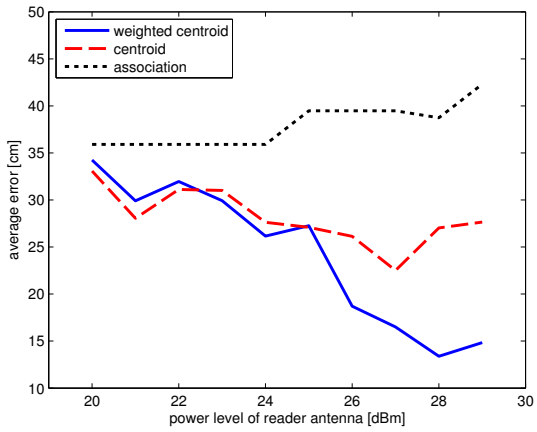


Figure 5.13: The effect of reader power on the average position error.

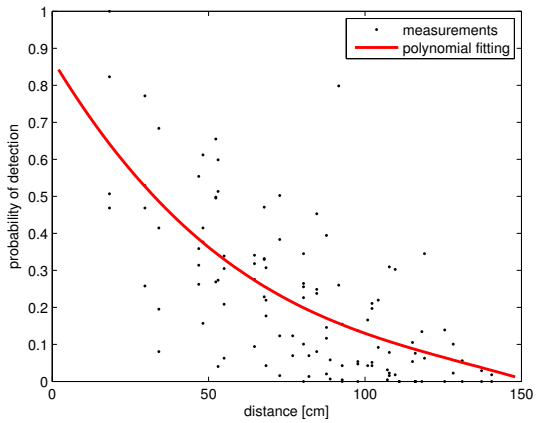


Figure 5.14: Estimated probability of detection and the corresponding four-degree polynomial fitting.

method. Clearly, with weighting the locations of the detected passive tags, we give higher emphasis to the detected tags that are closer than the ones that are further away from the sensatag.

For comparison of the performance of the methods, we also used the empirical CDF of the location error of the three methods. The results are shown in Figure 5.15. The CDFs of the errors confirm that the WC performs significantly better than the other two methods. For example, the probability of the error being less than 40cm is about 0.95 for the weighted centroid, 0.82 for the centroid, and 0.4 for the association-based method.

In the following set of experiments, we focus on WC method and test its robustness to environmental changes. Thus, we check the accuracy of the method for different orientation of the sensatag antenna, different positions of the reader antenna, and different LOS/NLOS scenarios. We can conclude the following:

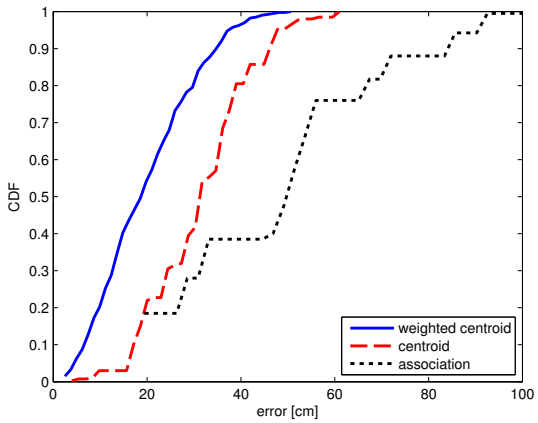
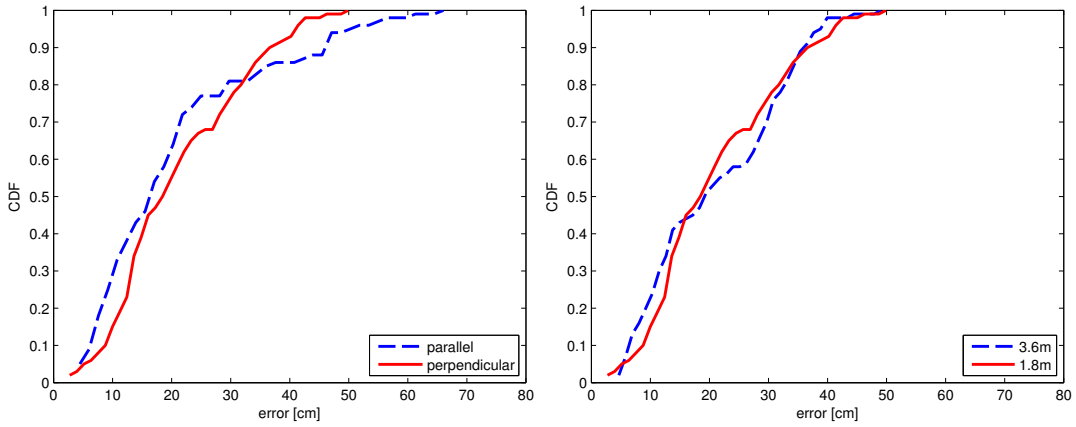


Figure 5.15: CDF of position error.



(a) (b)

Figure 5.16: CDF of position error for: (a) two different orientations of sensatag antenna (w.r.t. reader antenna), and (b) two different positions of the reader antenna.

- The orientation of the plane of sensatag antenna has significant impact on accuracy. Note that considered orientations represents the best and the worst case scenario. According to Figure 5.16a, the difference in error can be up to 15cm.
- Different distance between reader antenna and the center of deployment area affects slightly accuracy (Figure 5.16b). However, this is valid under assumption that reader antenna provides sufficient power level at whole deployment area (according to our tests, minimum power level should be -14 dBm).
- Only metal and fluid obstacles affects accuracy as expected (Figure 5.17). The wooden shelf practically has no impact.

Taking into account above conclusions, we can see that our method is robust to

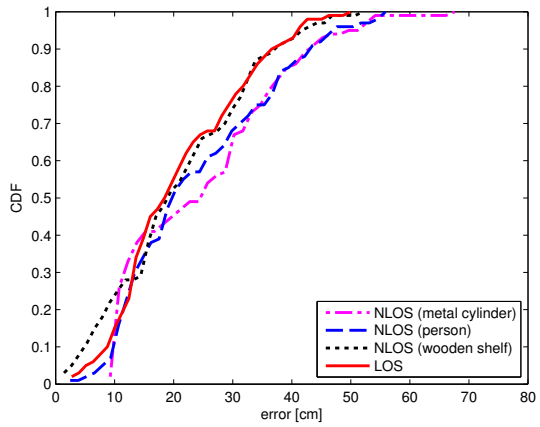


Figure 5.17: CDF of position error for LOS and three NLOS scenarios.

Figure 5.18: Experimental setup for sensatag tracking: There were 9 reference points (shown by the standing boxes), one reader (shown antenna in foreground), and one sensatag (on the chair) which represents the target.

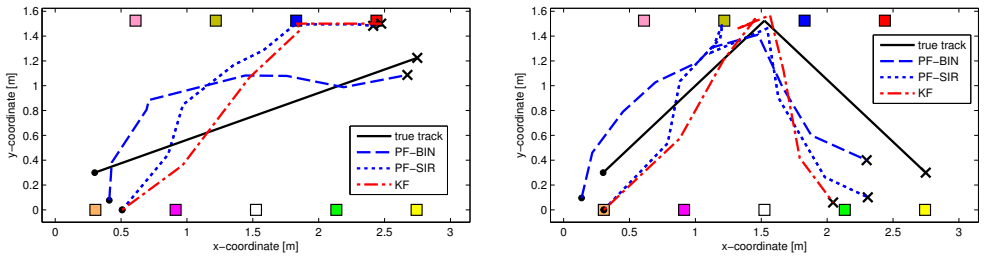
dynamic changes in the environment, so we expect that it can be used for a number of real applications. We confirmed it by performing the experiments for two typical applications (see Appendix C.2).

Sensatag tracking

Figure 5.18 shows our experimental setup. We deployed 9 reference tags in an area of 3m x 1.6m. The reader antenna was at a distance of about 2m from the center of the area, and its power level was set to 28dBm. The sensatag was placed on a chair with wheels that could be moved easily. Our objective was to track the sensatag during a period of 6s ($T_s \approx 0.7s$). In the experiment, the speed of the movement was approximately constant.

In the first set of experiments, our goal was to obtain calibration samples[6] used

[6]Due to the complex location protocol (see Section 5.3.1), sensatag-based RFID system was not fast enough to obtain the likelihood online.



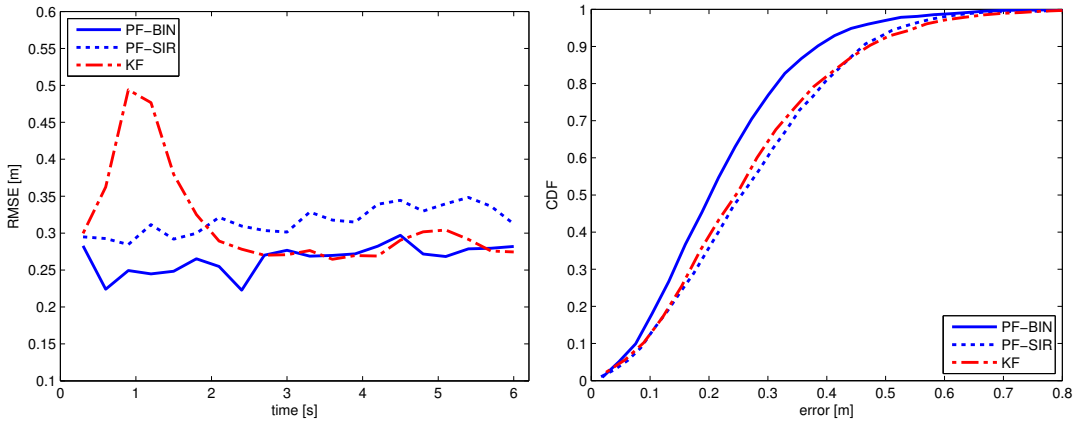
(a) (b)

Figure 5.19: Illustration of results of tracking for two different tracks: The starting point of the target is marked with a dot, and the destination point with an x.

for estimation of the likelihood (for PF-SIR), measurement covariance matrix (for KF) and probability of detection (for PF-BIN). To that end, we acquired 20 independent measurements at 20 grid points. Using these samples, we obtained empirical KDE of the measurement noise used for the PF-SIR method. For the KF method, we estimated the measurement covariance matrix $R = diag(0.025, 0.027)$, and assumed (without measuring) that the process covariance matrix was given by $Q = diag(0.2V, 0.2V)$ where V is the speed of movement. Finally, for the PF-BIN method, we estimated the parameters of our model for probability of detection as $\widehat{\alpha} = 3.059$ and $\widehat{d_0} = 0.32m$. Having defined all the parameters, we tracked the sensatag over a number of different tracks. We applied the KF, PF-SIR and PF-BIN methods. The results for two tracks are shown in Figure 5.19. Taking into account that the area is very small and that the measured data is very erroneous, obtained results are sufficiently accurate. Note also that the true track is also slightly uncertain because of the limited precision during movement. This is the reason why we are going to provide more precise comparison using simulations.

We conducted simulations of 100 random tracks. We used the same model, obtained from the real data. We changed the number of reference tags ($K = 16$), the sampling period ($T_s = 0.3$), and the deployment area (4m x 4m). The reference tags are placed in grid topology. According to Figure 5.20a, where we show the averaged RMSE over the 100 tracks, the PF-BIN consistently performed better than the PF-SIR. On the other hand, the KF had the worst performance during some initial period, probably because of the setup time that is necessary for parameter tuning. In Figure 5.20b, we plotted the CDF of the errors of each of the methods. As we can see, PF-BIN performed the best of all methods.

Regarding complexity, we found that PF-BIN was twice faster than the PF-SIR, but about 10 times slower than the KF. Thus, one may conclude that the KF is an option for a low-cost application where high accuracy is not crucial. However, if one wants to have a robust algorithm, PF-based methods should be applied. In our



(a) (b)

Figure 5.20: Comparison of the (a) RMSEs, and (b) CDFs of position errors.

experiments, we did not detect large outliers, but in general, they can be expected.

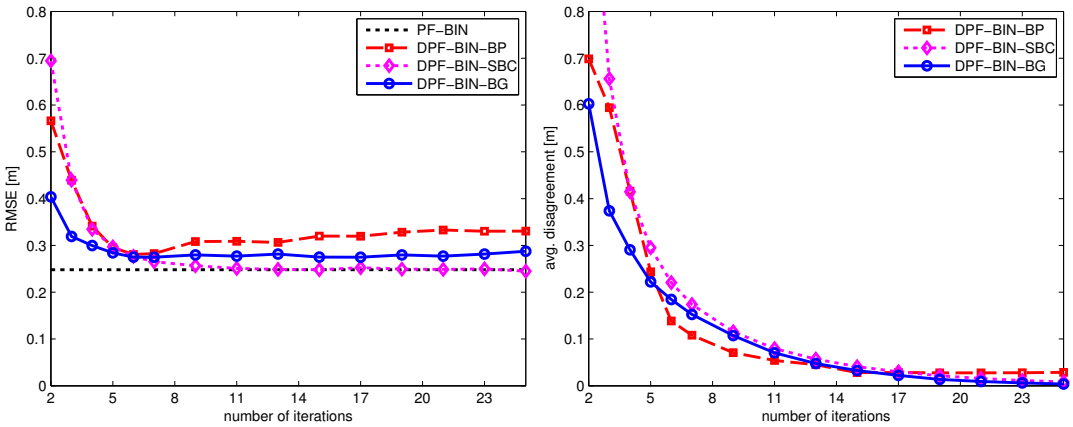
Distributed tracking

Our goal is to compare centralized PF-BIN method with three distributed algorithms proposed in Section 4.3.2: DPF-SBC, DPF-BG and DPF-BP. These methods will be also based on binomial likelihood function, so we denote them as: DPF-BIN-SBC, DPF-BIN-BG and DPF-BIN-BP. We reuse the model for probability of detection obtained from real data in previous experiments. We assume that there are 16 reference sensatags, the sampling period is $T_s = 0.3$, and the deployment area is 4m x 4m.

We additionally need to define the communication and sensing range of the sensatags. Taking into account model from Figure 5.11, sensatags can communicate with each other if the distance between them is less than $R \approx 1\text{m}$[7]. Regarding sensing range, according to Section 5.3.3, PF-BIN method has available observation even if none of the tags cannot be detected (i.e., this can be considered as negative information). Therefore, the sensing range is set to the diameter of the deployment area ($r = 4\sqrt{2}\text{m}$). Note that this is opposite to typical WSN (e.g., with previously used IRIS motes) in which the communication radius is larger than the sensing radius.

As a performance metric, we again use RMSE and average disagreement (defined in eq. (4.46) and (4.47)). The results are shown in Figure 5.21. As we can see, the behavior of DPF-BG and DPF-SBC is similar to theoretical results in Section 4.3.2.

[7]To ensure this radius, we recommend slightly larger power of the reader.



(a) (b)

Figure 5.21: Performance comparison of PF-BIN and DPF-BIN methods: (a) RMSEs, and (b) average disagreement.

However, DPF-BP provides slightly worse results in terms of disagreement, and slightly better results in terms of RMSE. We suppose that it is caused by different (binomial) likelihood function, and symmetric (grid) graph configuration.

5.4 Summary

In this chapter, we provided the experimental analysis of some of the proposed methods in previous chapters. We analyse cooperative localization based on NBP, and NBP-ST using Crossbow's IRIS wireless motes. Obtained results are very promising since they are very similar with the results based on theoretical models. Localization and tracking algorithms (including DPF-based distributed tracking) were analysed using sensatag-based RFID system. We also implemented single-node localization and tracking algorithms using RFID system in order to check their robustness to environmental changes and learn the appropriate models for simulations. They demonstrate that the tagged object can be localized and tracked with high accuracy.

Chapter 6

Conclusions and future work

6.1 Conslusions

We summarize below the main conclusions of this thesis:

- In Chapter 3, we proposed novel cooperative localization techniques based on nonparametric message passing methods. The first contribution is NBBP method which is capable to archive better performance than NBP with less particles. Nevertheless, the main contributions of this chapters are four novel algorithms for loopy networks: NGBP-JT, NGBP-PJT, NBP-ST, and URW-NBP. According to out results, we can conclude the following: i) NGBP-JT method, which provides accurate beliefs in loopy networks, has acceptable complexity only in small-scale sensor networks; i) NGBP-PJT can significantly outperform NBP, but it is not fully scalable, iii) NBP-ST can slightly outperform NBP method in highly-connected networks, and it is computationally feasible in large-scale ad-hoc/sensor networks; iv) URW-NBP is capable to run in asynchronous ad-hoc/sensor networks, and can slightly outperform NBP while keeping NBP's robustness to failures. Although all proposed methods can provide better estimates than NBP, our main conclusion is that URW-NBP leads to the best trade-off between accuracy, cost and robustness.

- In Chapter 4, we proposed novel algorithms for cooperative localization in mobile networks, and distributed target tracking. We extended NBP method for cooperative localization in mobile networks by providing: i) an optional 1-leg smoothing done almost in real-time, ii) a novel low-cost communication protocol, and iii) more efficient sampling techniques. Moreover, novel algorithms, PMC-NBP and ANBP, which use more efficient sampling techniques, consistently outperform standard NBP. For the distributed target tracking, we

proposed three DPF variants based on BC algorithms (DPF-SBC, DPF-BG, and DPF-BP). In contrast to previous methods, these methods can approximate global likelihood function in nonparametric form. According to our results, DPF-BG should be used in all tracking applications where minimal expected error is crucial. On the other hand, if the agreement of the estimates in the networks is more important than absolute error, DPF-BP could be a good choice. DPF-SBC, which guarantees convergence after large number of iterations, could be applied when the cost and latency are not crucial.

- In Chapter 5, we provided the experimental analysis of NBP, NBP-ST and DPF methods. We used Crossbow's IRIS wireless motes and novel semi-passive RFID system. We also implemented single-node localization and tracking algorithms using RFID system in order to check their robustness to environmental changes and learn the appropriate models for simulations. Obtained results are very promising since they are very similar with the results based on theoretical models.

6.2 Future work

Although we proposed a number of novel algorithms for cooperative localization and tracking, by no means, the study in this thesis is complete. In this section, we propose several interesting topics for the future research. They can be divided into four categories: i) cooperative localization in static networks, ii) cooperative localization in mobile networks, iii) distributed target tracking, and iv) experimental study of distributed algorithms.

Cooperative localization in static networks

One important remaining problem for the static networks is the case with non-rigid graphs, which usually appears in graphs with low connectivity. Although this problem was out of scope of this thesis, we know that bounded boxes and negative information can be very helpful (e.g., see Figure 3.3). However, we expect that additional improvements are possible. For example, if NBP run in collaborative subtrees (see Section 2.3.2), which are rigid, all the estimates will be unimodal. Then, postprocessing phase can take care of the nodes which are out of subtrees. Another interesting line of investigation could be additional improvements of the proposed algorithms in Chapter 3. Some tractable variant of TRW-NBP can be developed, e.g., by computing the edge appearance probabilities using only local node degree. Moreover, for GBP methods, it is eventually possible to find more

efficient approximations of the JT. Finally, some completely novel message-passing method can be developed, and compared with the proposed methods.

Cooperative localization in mobile networks

One line of investigation, in mobile networks, could be an additional improvement of the sampling techniques. For example, there is still question if it is possible to provide NBP with linear complexity in the number of particles, while keeping the similar performance. Moreover, one can apply the NBP and variations for mobile localization with more informative mobility model. In that case, it might not be necessary to run NBP in each time frame, so the algorithm can keep running even if NBP is relatively slow (comparing with sampling interval T_s). Regarding communication cost, one can find a different compression techniques for the packages that have to be transmitted. Finally, the security in sensor networks should be also investigated. The serious problems can happen in the whole network, if just one node transmits completely wrong message (a set of particles), or if an anchor node transmits wrong coordinates. We expect that these problems can be partially solved using the ideas from the game theory.

Distributed target tracking

The remaining problem of fast DPF algorithms (DPF-BG and DPF-BP) is no guarantee for the convergence to the true likelihood. DPF-BG provides the true likelihood only in expectation, and DPF-BP provides biased results in the networks with loops. One line of investigation could be a hybrid methods based on combination of SBC, BG and BP, which might provide better convergence/performance trade-off. For example, the method which runs DPF-BP couple of iterations, then switches to DPF-SBC, will likely inherit the fast convergence of DPF-BP and high accuracy of DPF-SBC. Moreover, DPF-BP can be also extended using ideas from Chapter 3. However, assuming that for this application, we prefer to avoid graph transformation (since for the mobile scenario we need low latency), DPF based on URW-BP could be a good option. Since all of the proposed algorithms are dedicated for single-target tracking (or tracking multiple targets which are well separated), one important line of investigation is multi-target tracking in which we need to estimate both the state and the label of the targets. Finally, distributed simultaneous localization and tracking (SLAT) in WSN could be very challenging topic. For this problem, the results from Chapter 3 (NBP-based cooperative localization), and Chapter 4 (DPF-based target tracking) can be combined.

Experimental study of distributed algorithms

Although this thesis already includes several real experiments in indoor environment, the algorithms are always run on the host computer (i.e., in centralized fashion). Therefore, it would be interesting to see how proposed algorithms perform if run, on each mote, in completely distributed way. Of course, we expect the similar accuracy as for already shown centralized emulations of distributed algorithms (in Chapter 5). However, the latency is still unknown. Thus, the main question is what is the minimal sampling interval (T_s) within which NBP algorithms and variations can be executed. The latency of ST and JT formations, which cannot be run in parallel, should be also analysed. These transformations of the graph also require a robust and synchronized protocol. Finally, distributed implementation can more precisely show us the level of robustness of all algorithms.

Appendix A

GBP-JT: example network

In this appendix, we analyze GBP-JT for small-scale networks illustrated in Figure A.1. The network has 10 nodes, 5 anchors (nodes 6-10) and 5 unknowns (nodes 1-5). There is a loop 1-2-4-5-3, so we have to triangulate it by adding two more edges (2-3 and 3-4). Then we define 8 cliques in the graph: $C_1 = \{x_1, x_2, x_3\}$, $C_2 = \{x_2, x_3, x_4\}$, $C_3 = \{x_3, x_4, x_5\}$, $C_4 = \{x_4, x_9\}$, $C_5 = \{x_5, x_{10}\}$, $C_6 = \{x_1, x_6\}$, $C_7 = \{x_2, x_7\}$, $C_8 = \{x_3, x_8\}$. The appropriate potentials of the three-node cliques are given by:

$$\begin{aligned} \psi_{C_1}(x_1, x_2, x_3) &= \psi_{12}(x_1, x_2)\psi_{13}(x_1, x_3) \\ \psi_{C_2}(x_2, x_3, x_4) &= \psi_{24}(x_2, x_4) \\ \psi_{C_3}(x_3, x_4, x_5) &= \psi_{35}(x_3, x_5)\psi_{45}(x_4, x_5) \end{aligned} \tag{A.1}$$

where we, for simplicity, assumed that single-node potentials are uninformative. Note that "virtual" edges do not appear in these equations since they are used only to define cliques. Other cliques, defined over pairs of nodes, represent the potential functions between two nodes (already known from standard BP):

$$\begin{aligned} &\psi_{C_4}(x_4, x_9) = \psi_{49}(x_4, x_9), \quad \psi_{C_5}(x_5, x_{10}) = \psi_{510}(x_5, x_{10}), \\ &\psi_{C_6}(x_1, x_6) = \psi_{16}(x_1, x_6), \quad \psi_{C_7}(x_2, x_7) = \psi_{27}(x_2, x_7), \\ &\psi_{C_8}(x_3, x_8) = \psi_{38}(x_3, x_8). \end{aligned} \tag{A.2}$$

The junction tree corresponding to the network in Figure A.1 is shown in Figure A.2. As we can see, "anchor cliques" ($C_4 - C_8$) do not receive the messages, so this graph does not contain loops. Actually, these "anchor cliques" also include one unknown node so we can send them messages, but this node can be also located by marginalizing the belief of some other clique.

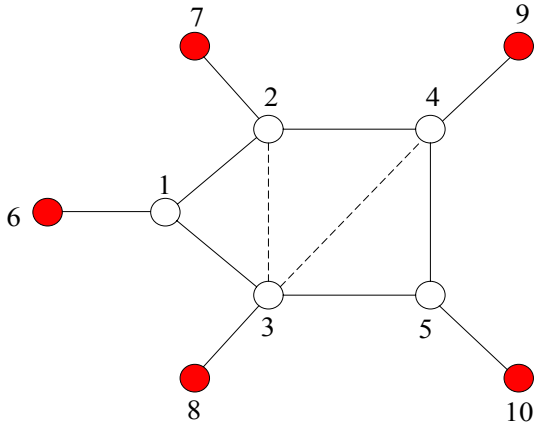


Figure A.1: Example of 10-node network with loop with 5 anchors (nodes 6-10), and 5 unknowns (nodes 1-5). The network is already triangulated by adding 2 more edges (marked by dashed lines)

In the next step, we can compute all messages using (3.18). The complete set of messages is given by:

$$m_{61}(x_1) = \psi_{16}(x_1, x_6^*), \quad m_{53}(x_5) = \psi_{510}(x_5, x_{10}^*) \tag{A.3}$$

$$m_{71}(x_2) = m_{72}(x_2) = \psi_{27}(x_2, x_7^*) \tag{A.4}$$

$$m_{42}(x_4) = m_{43}(x_4) = \psi_{49}(x_4, x_9^*) \tag{A.5}$$

$$m_{81}(x_3) = m_{82}(x_3) = m_{83}(x_3) = \psi_{38}(x_3, x_8^*) \tag{A.6}$$

$$m_{12}(x_2, x_3) = \psi_{27}(x_2, x_7^*)\psi_{38}(x_3, x_8^*) \sum_{x_1} \psi_{16}(x_1, x_6^*)\psi_{C_1} \tag{A.7}$$

$$m_{32}(x_3, x_4) = \psi_{49}(x_4, x_9^*)\psi_{38}(x_3, x_8^*) \sum_{x_5} \psi_{510}(x_5, x_{10}^*)\psi_{C_3} \tag{A.8}$$

$$m_{21}(x_2, x_3) = \psi_{27}(x_2, x_7^*)\psi_{38}(x_3, x_8^*) \sum_{x_4} \psi_{49}(x_4, x_9^*)\psi_{C_2} m_{32} \tag{A.9}$$

$$m_{23}(x_3, x_4) = \psi_{49}(x_4, x_9^*)\psi_{38}(x_3, x_8^*) \sum_{x_2} \psi_{27}(x_2, x_7^*)\psi_{C_2} m_{12} \tag{A.10}$$

where asterisk denotes the known location of the anchor node and the messages from "anchor cliques" are directly replaced by the appropriate potential function. Moreover, we used simplified notation for the messages and clique potentials on the right side of equations (e.g., $m_{12} = m_{12}(x_2, x_3)$, $\psi_{C_2} = \psi_{C_2}(x_2, x_3, x_4)$).

The beliefs of cliques are computed using (3.19):

$$M_1(x_1, x_2, x_3) = \psi_{C_1}\psi_{16}(x_1, x_6^*)\psi_{27}(x_2, x_7^*)\psi_{38}(x_3, x_8^*)m_{21} \tag{A.11}$$

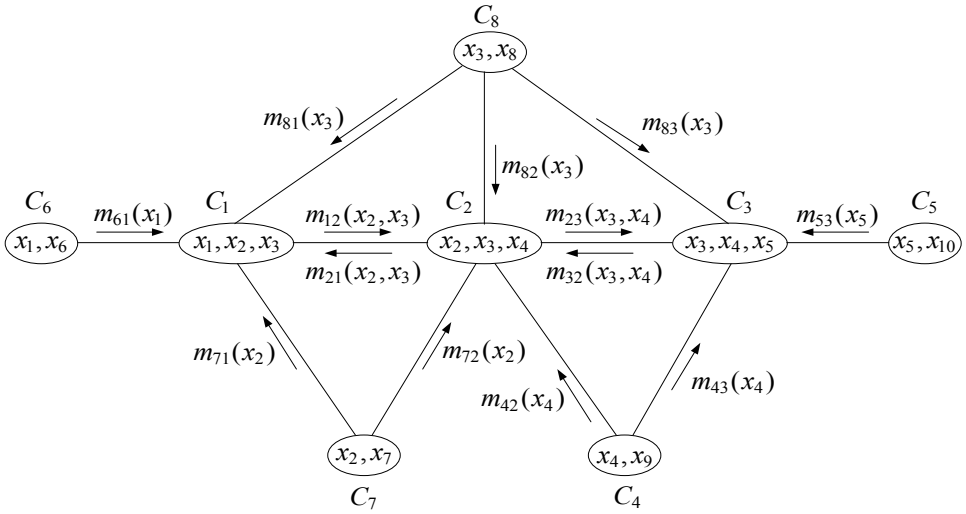


Figure A.2: The junction tree corresponding to the network in Figure A.1

$$M_2(x_2, x_3, x_4) = \psi_{C_2}\psi_{27}(x_2, x_7^*)\psi_{38}(x_3, x_8^*)\psi_{49}(x_4, x_9^*)m_{12}m_{32} \tag{A.12}$$

$$M_3(x_3, x_4, x_5) = \psi_{C_3}\psi_{38}(x_3, x_8^*)\psi_{49}(x_4, x_9^*)\psi_{510}(x_5, x_{10}^*)m_{23} \tag{A.13}$$

Now it is easy to compute beliefs of single nodes by marginalizing beliefs of cliques using (3.20). Obviously, it is sufficient to know beliefs of C_1 and C_3 since these cliques include all unknown nodes. Marginalization of C_2 provides a degree of freedom and could be used to check the estimated positions of some nodes (in our case, for the nodes 2, 3 and 4).

Appendix B

Convergence behavior of BP consensus

We analyse here the convergence behavior of BP consensus in loopy graphs. It is already well-known [77] that BP consensus (as a special case of standard BP) converges to the exact solution after a finite number of iterations in cycle-free graphs. Using an appropriate message schedule, this number of iterations is equal to $D_g + 1$, where D_g is the diameter of the graph (i.e., the maximum hop-distance between any two nodes). However, for general graphs, it is straightforward to show (using equation (4.40)) that the beliefs of BP consensus after $D_g + 1$ iterations is given by:

$$M_n^{(D_g+1)}(x_t) \propto \prod_{u \in G_t} p(y_{u,t}|x_t)^{\alpha_{u,n,t}} \tag{B.1}$$

where $\alpha_{u,n,t} \geq 1$ is an exponent ($\alpha_{u,n,t} \in \mathbb{N}$) of node pair (u, n) at time t. In case of cycle-free graphs $\alpha_{u,n,t} = 1$, so the estimated belief is equal to desired global likelihood (given by (4.26)). In case of $\alpha_{u,t,n} > 1$, the observation from node u at time t is *over-counted* at node n. To understand the overcounting behavior, we determine $\alpha_{\max}$, the maximum value (maximized over n and u) of $\alpha_{u,n,t}$ after $D_g + 1$ iterations. Note that running more than $D_g + 1$ iterations is unnecessary, as it will increase the α-values. While for the general case this problem is hard, we limit ourselves to some best- and the worst-case examples. In particular, we consider 4 representative graph configurations, shown in Figure B.1:

1) Fully-connected graph (clique): For the example in Figure B.1a, $D_g = 1$, so the belief at second iterations is given by (4.41). Since the graph is fully-connected, we know that the set G_n includes all nodes in the graph except node n (which is locally available). Therefore, $\alpha_{\max} = 1$, so BP consensus is correct.

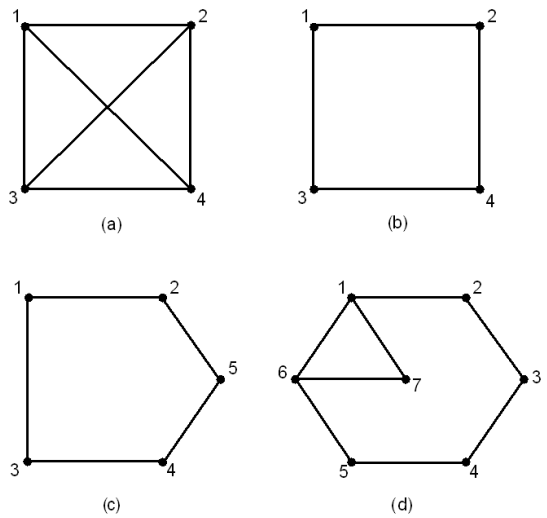


Figure B.1: Example graphs: (a) fully-connected graph ($D_g = 1$), (b) single-cycle graph with even number of nodes ($D_g = 2$), (c) single-cycle graph with odd number of nodes ($D_g = 2$), and (d) single-cycle graph with added short loop ($D_g = 3$).

2) Single-cycle graph with even number of nodes: For the example in Figure B.1b, $D_g = 2$, so we need to run 3 iterations of BP. In the second iteration, node 1 will obtain likelihood from nodes 2 and 3, but in the third iteration it will obtain likelihood from node 4 twice (through nodes 2 and 3). Therefore, $\alpha_{\max} = 2$.

3) Single-cycle graph with odd number of nodes: For the example in Figure B.1c, again $D_g = 2$, so we need to run 3 iterations of BP. In the second iteration, node 1 will obtain likelihood from nodes 2 and 3, and in the third iteration it will obtain likelihood from nodes 4 and 5. Therefore, $\alpha_{\max} = 1$, so BP consensus is correct.

4) Graph with short loops: For the example in Figure B.1d, $D_g = 3$, so we need 4 iterations of BP. After 4 iterations, nodes 1 and 6 will have triple-counted their own local likelihoods (since it has its own information, as well as messages received due to the clockwise and counter-clockwise circulation through short loop[1] 1-6-7). Therefore, $\alpha_{\max} = 3$. This reasoning can be generalized to a case with N_{sh} short loops (which all contain the edge 1-6), $\alpha_{\max} = 1 + 2N_{\text{sh}}$.

All previous claims can be easily proved by iterating (4.40). Taking into account that case 4) is the worst-case scenario, we can conclude that in the worst-case $\alpha_{\max} = 1+2N_{\text{sh}}$. This is not a promising conclusion, since $\alpha_{\max}$ can be unbounded, for fixed D_g, as the number of nodes grows. However, with good sensor deployment, highly asymmetrical configurations can be avoided.

[1]A short loop is defined as a loop that consists of 3 nodes.

Appendix C

Sensatag-based RFID localization

C.1 Functional blocks of the sensatag

In following text, we describe sensatag's main functional blocks: RF front end, analog section and digital section. A block diagram of the sensatag hardware is shown in Figure C.1.

RF Front End

The RF front end of the device consists of a passive envelope detector that is built using a Schottky Diode with corresponding matching circuit. When a passive RFID tag in the vicinity of the sensatag backscatters, the sensatag receives a signal that is a superposition of the tag backscatter and the continuous wave (CW) signal that the reader is transmitting during this time. The sensitivity of the sensatag to tags in its vicinity depends upon its ability to detect small changes in resultant power in this superimposed signal. This corresponds to the Δ RCS of the tag, i.e. the difference in tag antenna RCS when the tag backscatters a 1 vs when it backscatters a 0 . This means that the detector circuit needs to be optimized not for the maximum value of output voltage for a stated input power, but for maximum slope of the input power (P_{in}) vs output voltage (V_{out}) characteristic around the typical power levels of operation. This optimization was done by appropriately tuning the matching circuit and the time constant of load on the baseband side of the diode detector circuit.

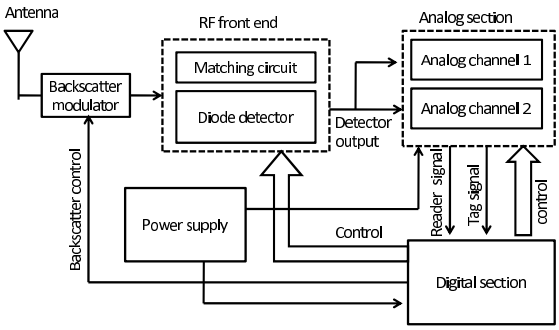


Figure C.1: Block diagram of the sensatag.

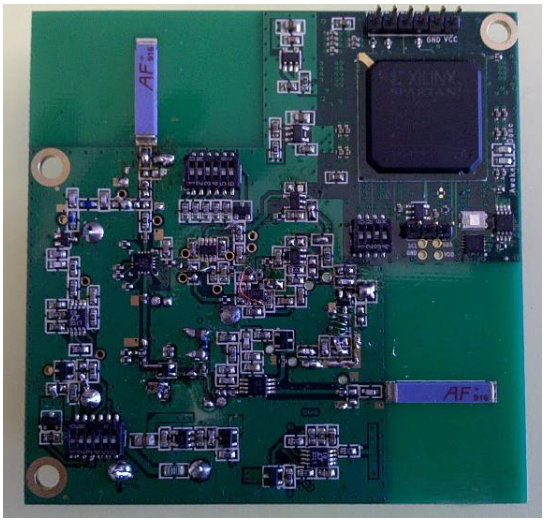


Figure C.2: Sensatag board used in the experiments

Analog Section

The sensatag analog section has the ability to process both the reader signal as well as the tag backscatter in order to produce a digital signal that can be processed by the digital section. The analog processing of the reader signal is exactly the same as in a standard passive tag. It consists of a buffer followed by a hysteresis comparator that generates the digital output. The processing of the tag backscatter is a bit more complex since the backscatter is a weak signal that has a significant DC offset due to the presence of the CW signal from the reader. The circuit consists of a band-pass filter (or a high-pass filter) for removing the DC offset, followed by a comparator that is configured as a data slicer. The filter parameters and the threshold generation circuit for the comparator are adaptive. This is achieved by changing the values of the RC components used in the circuit using switches controlled by the digital section.

Digital Section

The digital section runs the sensatag protocol and as such is the brain of the device. In the current version, the digital section is implemented on an FPGA platform. This platform is chosen to allow for rapid prototyping and verification of the digital section, particularly keeping in mind that ultimately, the sensatag will be implemented as an ASIC. The present embodiment uses a Xilinx Spartan 3AN FPGA. This device has an internal configuration memory which results in significant space saving on the digital section of the board. The current embodiment of the sensatag, used in the system described herein, is shown in Figure C.2.

C.2 Potential applications

Shelf identification

Passive tags are often used in warehouses, retail stores and offices for tagging a large number of items. Frequently, the tagged items are densely co-located. An RFID reader used in these situations detects a large number of tags in its field of view and is unable to determine the specific location of each tag. Sensatags can provide a very attractive solution for shelf-level localization of tagged items. It can be affixed to each shelf or bin location. The sensatag will be programmed with an identifier corresponding to the location to which it is attached. In addition, a Gen 2 UHF passive (or semi-passive) tag will be attached to each asset (box, case or pallet) stored in the shelf or bin. Inventory can be done by mobile readers which may be mounted on mobile carts, forklifts or in some cases may even be handheld. As the mobile reader moves through an isle, it reads the ID's of all the tagged items. Each sensatag senses backscattering tags in its own vicinity and reports this information to the reader in accordance with the protocol described in Section 5.3.1. Using the information obtained from the sensatags, the system can determine the shelf location of each item.

For sensatag-based RFID system, we perform an experiment in which we want to identify the shelf that has an item with an attached sensatag. The setup is shown in Figure C.3. In contrast to experiments in 5.3.4, the basis is vertical. In order to find on which shelf is the sensatag, we simply make quantization of previous location estimates (found by WC), i.e., if the location estimate is closest to center of the shelf j, we associate the sensatag with shelf j. We conducted 100 tests, and identified the correct shelf in all the trials.

Figure C.3: Experimental setup for typical warehouse application: The goal is to find on which shelf is sensatag.

Direction of movement (DOM) estimation

Accurate DOM estimation is crucial for many applications, in which we need to know DOM of the person or object close to some monitoring area. Storage facilities often inventory their assets as the enter and leave the facility by installing RFID portals at each entry/exit point. Conventional RFID systems suffer from two important shortcomings: i) ambiguity in direction of asset motion (entering or exiting the facility), and ii) cross reads between adjacent portals. These problems significantly hamper the business intelligence derived from the deployed RFID system. Sensatags can readily solve the above mentioned problems.

We describe here how to use location estimates for the DOM estimation. This problem can be easily solved if we use 2 antennas, by measuring the times of detection of the tagged object from the signals of the antennas [70]. However, due to the small physical distances of the whole setup, the more distant antenna from the incoming object would read the tag almost at the same time as the nearer antenna, which would prevent easy determination of the DOM. Our solution for estimating the DOM of the sensatag relies on a reader with one antenna and on a number of reference tags. In that case, DOM can be found by recognizing the pattern of the location samples (e.g., location estimates found by WC). One benefit of this method

is easy estimation of the non-movement of the tracked object, as explained below.

More specifically, our goal is to find if the target is moving left, moving right or standing within the monitoring area. Thus, for the problem at hand, the location estimates are one-dimensional (1D). Given two 1D estimates of the position at two consecutive instants, y_t and y_{t-1} (found by WC), we define the estimate of the direction:

$$dir = \begin{cases} '\text{move right}', & if\ y_t > y_{t-1} \\ '\text{stand}', & if\ y_t = y_{t-1} \\ '\text{move left}', & if\ y_t < y_{t-1} \end{cases} \tag{C.1}$$

Obviously, defined sample is not sufficient for reliable estimation of the direction because of the possible outliers, and limited precision (e.g., $y_t = y_{t-1}$ will practically never happen). One solution is to accumulate as many estimates dir as possible. Thus, we also define random variable n_r which represents the number of occurrences of $dir = '\text{move right}'$ in a set of N_s samples[1]. Assuming that $p(dir = '\text{move right}') = P_r(C)$ (where C is one of 3 possible classes: $C \in \{'\text{move right}, '\text{stand}', '\text{move left}'\}$)[2], n_r is distributed according to the binomial distribution:

$$N_r \sim Bin(n_r|N_s, P_r(C)) \tag{C.2}$$

Probabilities $P_r(C)$ can be obtained offline in appropriate environment of interest. Since the RFID system will provide N_s samples, from which we can easily estimate $n_r = N_r$, we can find the class by maximizing the likelihood:

$$\widehat{C} = \arg\max_C(Bin(N_r|N_s, P_r(C))) \tag{C.3}$$

It is already shown that this approach minimizes the misclassification rate [9]. We point out that we can also apply some other criteria, e.g., the maximization of a some specific utility function.

For the experiments, we used the similar setup as for localization (see Section 5.3.4). The number of reference points was 9 (4 placed on the left, and 5 placed on the right side of the area), and deployment area was 3m x 2m. In order to estimate probabilities $P_r(C)$, we moved the target sufficient number of times in appropriate direction, and measure the frequency of occurrence of $dir = '\text{move right}'$. Hence, we

[1]The results would be equivalent if we count $dir = '\text{move left}'$ samples.
[2]We assume that all other rare events (e.g., target circulating) are within the class 'stand'.

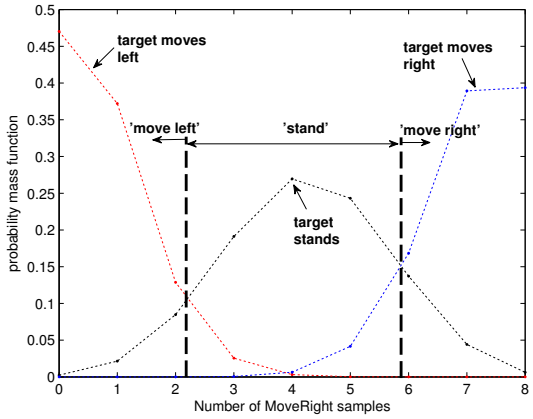


Figure C.4: Binomial distribution for all 3 classes, and corresponding decision bounds.

Table C.1: Accuracy of proposed method for DOM estimation

	'move right'	'stand'	'move left'
Target moves right	90%	10%	0%
Target stands	0%	95%	5%
Target moves left	0%	10%	90%

obtained:

$$P_r = \begin{cases} 0.89, & \text{if target moves right} \\ 0.53, & \text{if target stands} \\ 0.09, & \text{if target moves left} \end{cases} \tag{C.4}$$

We can see that one single estimate is not sufficient for reliable DOM estimation. Given these probabilities, we have defined class-conditional binomial distributions defined in (C.3). We illustrate it in Figure C.4, for the case of $N_s = 8$ obtained samples. As we can see, in order to minimize the misclassification rate, the decision bounds must be the intersections between appropriate binomials. Once we observe n_r, we simply choose the class which has the maximum probability.

In order to test this method, we performed experiments in which the target is moving left, right or standing. As we can see in Table C.1, we successfully estimated DOM in minimum 90% of the test cases (out of 60 tests). The misclassification, of course, cannot be avoided due to the overlap of the binomials, as shown in Figure C.4. We can reduce misclassification if N_s is larger so that the conditional distributions become tighter and with less overlap. This is not possible with current RFID system. However, it is important to note that, according to our results, proposed method never makes serious error, e.g., estimating that target moves left when it actually moves right.

Bibliography

[1] N. Ahmed, M. Rutten, T. Bessell, S. S. Kanhere, N. Gordon, and S. Jha, "Detection and tracking using particle-filter-based wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 9, pp. 1332–1345, Sept. 2010.

[2] M. S. Arulampalam, S. Maskell, N. G. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, Feb. 2002.

[3] AT86RF230, http://www.atmel.com/dyn/resources/prod_documents/doc5131.pdf.

[4] A. Athalye, V. Savic, M. Bolic, and P. M. Djuric, "A radio frequency identification system for accurate indoor localization," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 1777–1780.

[5] ATmega, http://www.atmel.com/dyn/resources/prod_documents/2549S.pdf.

[6] T. C. Aysal, M. E. Yildiz, A. D. Sarwate, and A. Scaglione, "Broadcast gossip algorithms for consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2748–2761, 2009.

[7] D. A. Bader and K. Madduri, "Designing multithreaded algorithms for breadth-first search and st-connectivity on the cray MTA-2," in *Proc. of Int. Conf. on Parallel Processing (ICPP)*, 2006, pp. 523–530.

[8] A. Baggio and K. Langendoen, "Monte Carlo localization for mobile wireless sensor networks," *Ad Hoc Networks*, vol. 6, no. 5, pp. 718–733, July 2008.

[9] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[10] M. Bolic, P. M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2442–2450, July 2005.

[11] M. Bolic, A. Athalye, and T. H. Li, *Performance of Passive UHF RFID Systems in Practice*. John Wiley & Sons, Ltd, 2010, pp. 1–22.

[12] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[13] Z. I. Botev, "A novel nonparametric density estimator," The University of Queensland, Australia, Tech. Rep., 2006.

[14] M. Bouet and A. L. dos Santos, "RFID tags: Positioning principles and localization techniques," in *Proc. of Wireless Days*, 2008, pp. 1–5.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, June 2006.

[16] M. Briers, A. Doucet, and S. S. Singh, "Sequential auxiliary particle belief propagation," in *Proc. of 8th Int. Conf. on Information Fusion*, vol. 1, 2005.

[17] M. F. Bugallo, M. Hong, and P. M. Djuric, "Marginalized population Monte Carlo," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2009, pp. 2925–2928.

[18] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing (WCMC)*, vol. 2, pp. 483–502, Sept. 2002.

[19] O. Cappe, A. Guillin, J.-M. Marin, C. P. Robert, and C. P. Roberty, "Population Monte Carlo," *Journal of Computational and Graphical Statistics*, vol. 13, pp. 907–929, 2004.

[20] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men, "Sequential Monte Carlo for simultaneous passive device-free tracking and sensor localization using received signal strength measurements," in *Proc. of IEEE/ACM Int. Conf. on Information Processing in Sensor Networks (IPSN)*, April 2011, pp. 342–353.

[21] C.-Y. Chong and S. P. Kumar, "Sensor networks: evolution, opportunities, and challenges," *Proc. of the IEEE*, vol. 91, no. 8, pp. 1247–1256, Aug. 2003.

[22] M. Coates, "Distributed particle filters for sensor networks," in *Proc. of 3rd Workshop on Information Processing in Sensor Networks (IPSN)*, April 2004, pp. 99–107.

[23] J. Costa, N. Patwari, and A. O. Hero, III, "Distributed multidimensional scaling with adaptive weighting for node localization in sensor networks," *ACM Trans. on Sensor Networks*, vol. -, pp. 1–26, June 2005.

[24] C. Crick and A. Pfeffer, "Loopy belief propagation as a basis for communication in sensor networks," in *Uncertainty in Artificial Intelligence*, 2003, pp. 159–166.

[25] Crossbow, http://www.xbow.com.

[26] K. Das and H. Wymeersch, "Censored cooperative positioning for dense wireless networks," in *Proc. of IEEE 21st Int. Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2010, pp. 262–266.

[27] A. D. G. Dimakis, A. D. Sarwate, and M. J. Wainwright, "Geographic gossip: Efficient averaging for sensor networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 3, pp. 1205–1216, March 2008.

[28] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. of the IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.

[29] P. M. Djuric, J. Beaudeau, and M. Bugallo, "Non-centralized target tracking with mobile agents," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 5928–5931.

[30] P. M. Djuric, M. Vemula, and M. F. Bugallo, "Target tracking by particle filtering in binary sensor networks," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2229–2238, June 2008.

[31] P. Djuric and A. Athalye, "RFID system and method for localizing and tracking a moving object with an rfid tag," US Patent 11 799 257, 2010.

[32] P. Djuric, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Processing Magazine*, vol. 20, no. 5, pp. 19–38, Sept. 2003.

[33] D. M. Dobkin, *The RF in RFID: Passive UHF RFID in Practice*. Newton, MA, USA: Newnes, 2007.

[34] N. I. Dopico, B. Bejar, S. V. Macua, P. Belanovic, and S. Zazo, "Improved animal tracking algorithms using distributed Kalman-based filters," in *European Wireless Conference*, 2011, pp. 1–8.

[35] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: fifteen years later," The Oxford Handbook of Nonlinear Filtering, 2011.

[36] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo localization: efficient position estimation for mobile robots," in *Proc. of the National Conference on Artificial Intelligence*, Orlando, Florida, 1999.

[37] B. Frey and D. MacKay, "A revolution: Belief propagation in graphs with cycles," in *In Neural Information Processing Systems.* MIT Press, 1998, pp. 479–485.

[38] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem, "Distributed online localization in sensor networks using a moving target," in *Proc. of 3rd Int. Symp. on Information Processing in Sensor Networks (IPSN)*, April 2004, pp. 61–70.

[39] A. F. Garcia-Fernandez and J. Grajal, "Multitarget tracking using the joint multitrack probability density," in *Proc. of 12th Int. Conf. on Information Fusion*, 2009, pp. 595–602.

[40] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 70–84, July 2005.

[41] D. Gu, "Distributed particle filter for target tracking," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, April 2007, pp. 3856–3861.

[42] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in *Proc. of Int. Conf. Information and Automation*, 2008, pp. 302–307.

[43] F. Gustafsson and F. Gunnarsson, "Mobile positioning using wireless networks: possibilities and fundamental limitations based on available wireless network measurements," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 41–53, July 2005.

[44] O. Hlinka, O. Sluciak, F. Hlawatsch, P. M. Djuric, and M. Rupp, "Distributed gaussian particle filtering using likelihood consensus," in *Proc. of IEEE Int.*

Conf. on Acoustics, Speech and Signal Processing (ICASSP), May 2011, pp. 3756–3759.

[45] F. Hutter, B. Ng, and R. Dearden, "Incremental thin junction trees for dynamic bayesian networks," Darmstadt University of Technology, Tech. Rep., 2004.

[46] A. T. Ihler, J. W. I. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, April 2005.

[47] A. T. Ihler, E. B. Sudderth, W. T. Freeman, and A. S. Willsky, "Efficient multiscale sampling from products of Gaussian mixtures," in *Proc. of Neural Information Processing Systems (NIPS)*. Cambridge, MA: MIT Press, 2003.

[48] A. T. Ihler, "Inference in sensor networks: Graphical models and particle methods," Ph.D. dissertation, MIT, 2005.

[49] H. Jamali Rad, A. Amar, and G. Leus, "Cooperative mobile network localization via subspace tracking," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 2612–2615.

[50] X. Ji and H. Zha, "Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling," in *Proc. of IEEE INFOCOM*, vol. 4, 2004, pp. 2652–2661.

[51] B. Jiang and B. Ravindran, "Completely distributed particle filters for target tracking in sensor networks," in *Proc. of IEEE Int. Parallel & Distributed Processing Symp.*, 2011, pp. 334–344.

[52] M. Jordan and Y. Weiss, "Graphical models: probabilistic inference," *Handbook of Neural Networks and Brain Theory. 2nd edition*, vol. -, pp. 1–17, 2002.

[53] W. Kim, K. Mechitov, J.-Y. Choi, and S. Ham, "On target tracking with binary proximity sensors," in *Proc. of 4th Int. Symp. on Information Processing in Sensor Networks (IPSN)*, 2005, pp. 301–308.

[54] C. Knapp and G. Carter, "The generalized correlation method for estimation of time delay," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 4, pp. 320–327, 1976.

[55] D. Koller and N. Friedman, *Probabilistic graphical models*. MIT Press, 2009.

[56] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1568–1583, 2006.

[57] J. H. Kotecha and P. M. Djuric, "Gaussian sum particle filtering," *IEEE Transactions on Signal Processing*, vol. 51, no. 10, pp. 2602–1612, Oct. 2003.

[58] C. M. Kreucher, "An information-based approach to sensor resource allocation," Ph.D. dissertation, The University of Michigan, 2005.

[59] K. Krishna and M. Narasimha Murty, "Genetic K-means algorithm," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 3, pp. 433–439, 1999.

[60] P. Krishnan, A. S. Krishnakumar, W.-H. Ju, C. Mallows, and S. N. Gamt, "A system for LEASE: location estimation assisted by stationary emitters for indoor rf wireless networks," in *Proc. of IEEE INFOCOM*, vol. 2, 2004, pp. 1001–1011.

[61] K. Langendoen and N. Reijers, "Distributed localization in wireless sensor networks: a quantitative comparison," *Elsevier Computer Networks*, vol. 43, no. 4, pp. 499–518, Nov. 2003.

[62] S. H. Lee and M. West, "Markov chain distributed particle filters (MCDPF)," in *Proc. of 48th IEEE Conf. held jointly with the 2009 28th Chinese Control Conf Decision and Control (CDC/CCC)*, 2009, pp. 5496–5501.

[63] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Comput. Networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[64] J. Miguez, "Analysis of parallelizable resampling algorithms for particle filtering," *Signal Processing*, vol. 87, no. 12, pp. 3155 – 3174, 2007.

[65] A. J. Mooij, N. Goga, and W. Wesselink, "A distributed spanning tree algorithm for topology-aware networks," in *Proc. of the Conf. on Design, Analysis, and Simulation of Distributed Systems*, 2004.

[66] J. M. Mooij and H. J. Kappen, "Sufficient conditions for convergence of the sum–product algorithm," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4422–4437, Dec. 2007.

[67] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "LANDMARC: indoor location sensing using active RFID," in *Proc. of 1st IEEE Int. Conf. on Pervasive Computing and Communications (PerCom)*, 2003, pp. 407–415.

[68] D. Niculescu and B. Nath, "Ad hoc positioning system (APS)," in *Proc. of IEEE GLOBECOM*, vol. 5, 2001, pp. 2926–2931.

[69] ——, "Ad hoc positioning system (APS) using AOA," in *Proc. of IEEE INFOCOM*, vol. 3, Mar./Apr. 2003, pp. 1734–1743.

[70] Y. Oikawa, "Tag movement direction estimation methods in an RFID gate system," in *Proc. of 6th Int. Symp. on Wireless Communication Systems (ISWCS)*, 2009, pp. 41–45.

[71] A. Oka and L. Lampe, "Distributed target tracking using signal strength measurements by a wireless sensor network," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 7, pp. 1006–1015, Sept. 2010.

[72] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520 – 1533, Sept. 2004.

[73] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. S. Shamma, "Belief consensus and distributed hypothesis testing in sensor networks," in *Proc. of NESC Workshop*. Springer Verlag, 2006, pp. 169–182.

[74] B. N. Oreshkin and M. J. Coates, "Asynchronous distributed particle filter via decentralized evaluation of gaussian products," in *Proc. of 13th Conf. on Information Fusion (FUSION)*, 2010, pp. 1–8.

[75] B. W. Parkinson and J. J. Spilker, *Global Positioning System: Theory & Applications (Volume One)*. Progress in Astronautics and Aeronautics, 1996.

[76] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, III, R. L. Moses, and N. S. Correal, "Locating the nodes: cooperative localization in wireless sensor networks," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 54–69, July 2005.

[77] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo: Morgan Kaufmann, 1988.

[78] R. Peng and M. L. Sichitiu, "Robust, probabilistic, constraint-based localization for wireless sensor networks," in *Proc. of 2nd IEEE Conf. on Sensor and Ad Hoc Communications and Networks (SECON)*, 2005, pp. 541–550.

[79] F. Penna, H. Wymeersch, and V. Savic, "Uniformly reweighted belief propagation for distributed bayesian hypothesis testing," in *Proc. of IEEE Statistical Signal Processing Workshop (SSP)*, June 2011, pp. 733 –736.

[80] T.-D. Pham, H. Q. Ngo, V.-D. Le, S. Lee, and Y.-K. Lee, "Broadcast gossip based distributed hypothesis testing in wireless sensor networks," in *Proc. of Int. Conf. on Advanced Technologies for Communications*, 2009, pp. 84–87.

[81] M. K. Pitt and N. Shephard, "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, vol. 94, no. 446, pp. 590–599, June 1999.

[82] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," MIT Laboratory for Computer Science, Tech. Rep., 2003.

[83] T. S. Rappaport, J. H. Reed, and B. D. Woerner, "Position location using wireless communications on highways of the future," *IEEE Communications Magazine*, vol. 34, no. 10, pp. 33–41, 1996.

[84] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995.

[85] K. Romer, "The lighthouse location system for smart dust," in *Proc. of the 1st int. conf. on Mobile systems, applications and services*, ser. MobiSys '03, 2003, pp. 15–30.

[86] T. Sanpechuda and L. Kovavisaruch, "A review of RFID localization: Applications and techniques," in *Proc. of 5th Int. Conf. on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, vol. 2, may 2008, pp. 769 –772.

[87] V. Savic, A. Poblacion, S. Zazo, and M. Garcia, "An experimental study of RSS-based indoor localization using nonparametric belief propagation based on spanning trees," in *Proc. of 4th Int. Conf. on Sensor Technologies and Applications (SENSORCOMM)*, July 2010, pp. 238–243.

[88] ——, "Indoor positioning using nonparametric belief propagation based on spanning trees," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, pp. 1–12, Aug. 2010.

[89] V. Savic and S. Zazo, "Nonparametric boxed belief propagation for localization in wireless sensor networks," in *Proc. of 3rd Int. Conf. on Sensor Technologies and Applications (SENSORCOMM)*, 2009, pp. 520–525.

[90] ——, "Sensor localization using nonparametric generalized belief propagation in network with loops," in *Proc. of 12th Int. Conf. on Information Fusion*, 2009, pp. 1966–1973.

[91] ——, "Nonparametric belief propagation based on spanning trees for cooperative localization in wireless sensor networks," in *Proc. IEEE 72nd Vehicular Technology Conf. Fall (VTC-Fall)*, Sept. 2010, pp. 1–5.

[92] ——, "Pseudo-junction tree method for cooperative localization in wireless sensor networks," in *Proc. of 13th Int. Conf. on Information Fusion*, July 2010, pp. 1–8.

[93] ——, "Belief propagation techniques for cooperative localization in wireless sensor networks," *in Handbook of Position Location: Theory, Practice, and Advances (eds S. A. Zekavat and R. M. Buehrer)*, Wiley, 2011.

[94] V. Savic, A. Athalye, M. Bolic, and P. M. Djuric, "Particle filtering for indoor RFID tag tracking," in *Proc. of IEEE Statistical Signal Processing Workshop (SSP)*, June 2011, pp. 193 –196.

[95] V. Savic, H. Wymeersch, F. Penna, and S. Zazo, "Optimized edge appearance probability for cooperative localization based on tree-reweighted nonparametric belief propagation," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3028–3031.

[96] V. Savic and S. Zazo, "Sensor localization using generalized belief propagation in network with loops," in *Proc. of the 17th European Signal Processing Conference (EUSIPCO)*, 2009.

[97] A. Savvides, H. Park, and M. B. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Proc. of the 1st ACM international workshop on Wireless sensor networks and applications*. ACM, 2002, pp. 112–121.

[98] A. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location: challenges faced in developing techniques for accurate wireless location information," *IEEE Signal Processing Magazine*, vol. 22, no. 4, pp. 24–40, July 2005.

[99] J. Schiff, E. B. Sudderth, and K. Goldberg, "Nonparametric belief propagation for distributed tracking of robot networks with noisy inter-distance measurements," in *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 1369–1376.

[100] Y. Shang, W. Rumi, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 15, no. 11, pp. 961–974, Nov. 2004.

[101] Y. Shang and W. Ruml, "Improved MDS-based localization," in *Proc. of IEEE INFOCOM*, vol. 4, 2004, pp. 2640–2651.

[102] Y. Shen, H. Wymeersch, and M. Z. Win, "Fundamental limits of wideband localization— part II: Cooperative networks," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 4981–5000, 2010.

[103] X. Sheng, Y.-H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," in *Proc. of Fourth Int. Symp. Information Processing in Sensor Networks (IPSN)*, 2005, pp. 181–188.

[104] B. Silverman, *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, New York, 1986.

[105] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky, "Nonparametric belief propagation," in *Proc. of IEEE Int. Conf. on Computer Vision and Pattern Recognition*, vol. 1, 2003.

[106] D. Ustebay, M. Coates, and M. Rabbat, "Distributed auxiliary particle filters using selective gossip," in *Proc. of IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 3296–3299.

[107] R. van der Merwe, A. Doucet, N. D. Freitas, and E. Wan, "The unscented particle filter," in *Proc. of Advances in Neural Information Processing Systems*, Nov. 2001.

[108] M. Vemula, M. F. Bugallo, and P. M. Djuric, "Sensor self-localization with beacon position uncertainty," *Signal Processing*, vol. 89, pp. 1144–1154, June 2009.

[109] V. Vivekanandan and V. W. S. Wong, "Concentric anchor beacon localization algorithm for wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 5, pp. 2733–2744, Sept. 2007.

[110] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky, "Tree-reweighted belief propagation algorithms and approximate ml estimation by pseudo-moment matching," in *Proc. of AISTATS*, 2003.

[111] M. Wainwright and M. Jordan, *Graphical models, exponential families, and variational inference*. Now Publishers Inc., 2008.

[112] G. Wang and K. Yang, "A new approach to sensor node localization using RSS measurements in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 10, no. 5, pp. 1389–1395, May 2011.

[113] X. Wang, X. Wang, and D. M. Wilkes, "A divide-and-conquer approach for minimum spanning tree-based clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 7, pp. 945–958, 2009.

[114] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural Computation*, vol. 12, pp. 1–41, 2000.

[115] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in Gaussian graphical models of arbitrary topology," *Neural Computation*, vol. 13, pp. 2173–2200, 2001.

[116] G. Welch and G. Bishop, "An introduction to the Kalman filter," University of North Carolina at Chapel Hill, Tech. Rep., July 2006.

[117] J. L. Williams and R. A. Lau, "Data association by loopy belief propagation," in *Proc. of 13th Conf. on Information Fusion*, July 2010, pp. 1–8.

[118] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," *Proc. of the IEEE*, vol. 97, no. 2, pp. 427–450, 2009.

[119] H. Wymeersch, F. Penna, and V. Savic, "Uniformly reweighted belief propagation for estimation and detection in wireless networks," *IEEE Transactions on Wireless Communications*, pp. 1–9, 2012.

[120] ——, "Uniformly reweighted belief propagation: A factor graph approach," in *Proc. of IEEE Int. Symp. on Information Theory (ISIT)*, July 2011.

[121] J. S. Yedidia, W. T. Freeman, and Y. Weiss, *Understanding belief propagation and its generalizations.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 239–269.

[122] J. Yi, S. Yang, and H. Cha, "Multi-hop-based Monte Carlo localization for mobile sensor networks," in *Proc. of 4th IEEE Conf. on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, 2007, pp. 162–171.

[123] A. Yoo, E. Chow, K. Henderson, W. McLendon, B. Hendrickson, and U. Catalyurek, "A scalable distributed parallel breadth-first search algorithm on bluegene/L," in *Proc. ACM/IEEE SC 2005 Conf. Supercomputing*, 2005.

[124] Y. Zhao, Y. Liu, and L. M. Ni, "VIRE: Active RFID-based localization using virtual reference elimination," in *Proc. of Int. Conf. on Parallel Processing (ICPP)*, 2007.